2003年度修士論文

レーザースキャナを用いた
地すべり地形変位観測のための三次元モデリング

Three-dimensional Modeling
for Landslide Displacement Monitoring by Laser Scanner

2004年1月

指導教員　高木　方隆

高知工科大学大学院工学研究科基盤工学専攻

社会システム工学コース　1065090

光岡　操

論文要旨

地すべりとは、斜面の土塊が移動する現象である。この地すべりによる災害を未然に防ぐためには地すべり全体の挙動把握が重要である。現在、地すべりの挙動把握は孔内傾斜計や伸縮計、GPS などを用いて行なわれている。しかし、これらの計測方法は点の計測である。地すべり全体の挙動把握を行なうには、高精度時系列標高モデルを生成する必要がある。

近年レーザースキャナ計測は、瞬時に数百万点もの点を取得できることから高精度な三次元モデル生成のために使用されている。しかし、レーザースキャナデータは膨大なデータ量となるため、市販されている CAD や CG、GIS ソフトでは対応困難である。そのため、レーザースキャナデータはオリジナルプログラムにより扱わなければならない。

本研究の目的は、レーザースキャナにより時系列で地すべりの計測を行ない、地すべりの挙動把握を行なうことである。まず、グリッドタイプの数値標高モデル、等高線、断面図、サーフェイスモデルの作成を行なう。次に、オリジナルプログラムを用いて、各三次元モデルを用いた変位抽出を行なう。最終的には、変位抽出システムの構築を行なう。本研究では、高知県仁淀村長者地すべりを対象とした。

地すべり地の測定は、2003 年 7 月と 2003 年 9 月に行なった。レーザースキャナデータからの変位抽出は、オリジナルプログラムを用いることにより抽出することができた。個々の三次元モデルを使用した変位抽出結果を以下にまとめる。

- 数値標高モデル：信頼性は、グリッド間隔に依存し、他のモデルよりも低い。しかし、容易に変位抽出を行なうことが可能である。

- 等高線・断面図：等高線や断面の形状の変化を見ることができるが、全体的な変位を求めるにすぎない。そのため、大まかな変位箇所を抽出した後に行なうことが望ましい。

- サーフェイス：対象物が大きく近い場所より計測可能でるとき、対象物の自動抽出による変位量ベクトル化により高精度で絶対座標での変位抽出を行なうことが可能である。しかし、対象物の自動抽出の信頼性は抽出面の大きさと方向に影響されると考えられ、今回の対象物では困難であった。

本研究の結果より、以下のような変位抽出システムの提案をする。

01）数値標高モデルにより、X、Y、Z 方向の変位抽出を行なう。

02）等高線・断面図により、01）で変位が見られた箇所について変位抽出を行なう。

03）同一面抽出可能な対象物が存在する場合、対象物の自動抽出による変位量ベクトル化を行ない、変位抽出を行なう。

Abstract

Landslide is phenomenon of mass movement in terrain. In order to prevent landslide, understanding behavior of landslide is important. The behavior of landslide is usually measured using extensometer, inclinometer or GPS (Global Positioning System). However, those are equipments for just point measurement. In this situation, understanding the behavior in whole landslide requires very high-resolution time-series three-dimensional elevation model.

Nowadays, Laser Scanner can be used for generating high-resolution three-dimensional model, which can easily measure millions of points. However, data volume of Laser Scanner is so huge that general commercial software (CAD, CG and GIS) cannot apply. So, Laser Scanner data must be accessed by original program.

Objectives of this study are measuring landslide by Laser Scanner in time-series and detecting the displacements. Firstly, three-dimensional elevation model including grid type model, contour line, cross section line and surface model are generated. Next, detecting software of landslide displacement in each three-dimensional model is programmed by C language. Finally, landslide displacement monitoring system will be developed. In this study, Tyojya-landlide, Niyodo-mura, Kochi was selected for test area.

Measurements of landslide using Laser Scanner were carried out in July, September and December 2003. The landslide displacements from the laser scanner data were successfully detected by original C programs. The results of the detecting landslide displacements in each three-dimensional elevation model were explained as follows ;

- Grid type : displacements monitoring was very easy to detect though accuracy was lower than other models. The accuracy was depending on grid size.
- Contour line and Cross section line : displacement monitoring was also easy in high accuracy after finding displacement area and direction.
- Surface model : displacements monitoring was done in high accuracy when extracting same surface in different period. However it was very difficult to extract the same surface automatically. The accuracy was depending on area size and direction of the surface. Finally, following displacement monitoring system was developed.

01) Displacements extracted using grid type model in X, Y, Z direction.
02) When displacements are found, contour line and cross section line were generated.
03) The high accuracy displacements monitoring based on surface model is carried out when surface extraction is succeeded.

――――――――――――――――――――目次――――――――――――――――――――

1. はじめに……1
 1-1. 地すべりについて
 1-2. レーザースキャナデータ
 1-3. 市販のソフトウェアについて
2. 目的……2
3. 現地概要……4
 3-1. 位置
 3-2. 地形
 3-3. 地質
 3-4. 対策工と地すべりの動き
4. レーザースキャナ……5
 4-1. 概要
 4-2. 計測原理
 4-3. 使用したレーザースキャナ
5. データの取得……7
 5-1. 対象・位置
 5-2. 取得データ
6. 数値標高モデル……9
 6-1. 数値標高モデルとは
 6-2. 作成アルゴリズムフロー
 6-3. 変位抽出
7. 等高線……12
 7-1. 等高線とは
 7-2. 作成アルゴリズムフロー
 7-3. 変位抽出
8. 断面図……15
 8-1. 断面図とは
 8-2. 作成アルゴリズムフロー
 8-3. 変位抽出
9. サーフェイスモデル……18
 9-1. サーフェイスモデルとは
 9-2. 作成アルゴリズムフロー
 9-3. 変位抽出
  9-3-1. グリッド上でのサーフェイス演算
  9-3-2. 最近隣点探索による
   変位量ベクトル化
  9-3-3. 対象物の自動抽出による
   変位量ベクトル化
10. 考察……25
11. 参考文献……28
12. おわりに……29
13. 謝辞……30
14. 付録……31
 A. 使用 PC スペック
 B. 作成プログラム
  B-01. 基準点計測データを座標変換
  B-02. 生データをテキストデータ変換
  B-03. 回転行列係数を用いた座標変換
  B-04. 数値標高モデル作成
  B-05. 等高線作成
  B-06. 断面図作成
  B-07. サーフェイスモデル作成
  B-08. 画像演算
  B-09. 等高線変位量計算
  B-10. 断面図変位量計算
  B-11. グリッド上のサーフェイス演算
  B-12. 最近隣点探索によるベクトル化
  B-13. 対象物の自動抽出

―――――――――――――――――――図目次―――――――――――――――――――

図 3.1 長者地すべりの位置図……4
図 3.2 長者地すべりの全景……4
図 4.1 レーザースキャナの計測原理……5
図 4.2 LMS-Z210 のスキャニング方向と軸と角度の定義図……7
図 5.1 計測対象……7
図 5.2 1 時期目の使用データ範囲と基準点位置（反射強度画像）……8
図 5.3 2 時期目の使用データ範囲と基準点位置（反射強度画像）……8
図 6.1 作成した 10㎝ グリッドの数値標高モデル（左：1 時期目、右：2 時期目）……9
図 6.2 10㎝ グリッドでの変位抽出結果画像……11
図 7.1 作成した等高線（左：1 時期目、右：2 時期目）……12
図 7.2 2 時期分の等高線を重ねたもの（青線：1 時期目、赤線：2 時期目）……14
図 7.3 等高線を用いた変位量計算イメージ図……14
図 8.1 作成した断面図（左：1 時期目、右：2 時期目）……15
図 8.2 2 時期分の断面図を重ねたもの（青線：1 時期目、赤線：2 時期目）……17
図 9.1 作成したサーフェイスモデル（1 時期目：約 100 ライン）……18
図 9.2 グリッド上でのサーフェイス演算方法のイメージ図（緑線：変位量）……20
図 9.3 変位算出方向を Y としたときのサーフェイス探索イメージ図……20
図 9.4 グリッド上でのサーフェイス演算結果（10㎝ グリッド,1㎝ オーダーの変位量）……21
図 9.5 図 9.4 のヒストグラム……21
図 9.6 最近隣点探索による変位量ベクトル化（約 100 ラインの下段 3 段部分）……22
図 9.7 対象物の自動抽出イメージ図……23
図 9.8 対象物の自動抽出結果（下流側から 42 ブロック分）……23
図 9.9 面抽出による変位量ベクトル化（下流側から 42 ブロック分）……24

―――――――――――――――――――表目次―――――――――――――――――――

表 1.1 野村らの用いたデジタルカメラの内部標定要素……2
表 1.2 野村らの用いた PC の性能……2
表 1.3 写真測量とレーザースキャナ計測の比較（カタログ値）……2
表 4.1 中～長距離計測用の地上型レーザースキャナの種類（カタログ値）……6
表 4.2 LMS-Z210 距離測定器性能（カタログ値）……6
表 4.3 LMS-Z210 スキャニング性能（カタログ値）……6
表 4.4 1 回の計測で取得できるデータの諸元……6
表 A-1 本研究で用いた PC の性能……31

1. はじめに

1-1. 地すべりについて

地すべりとは、雨や地下水などの影響により、斜面を構成する地山の内部において力学的なバランスが何らかの原因によって破壊され、地中に発生した破壊面を境としてそれよりも上側に存在する斜面構成物質が重力の作用により連続的または間欠的に比較的緩慢な速度（0.01～10㎜/day 程度）で移動する現象を言う。地すべり地においては、地表に亀裂や段差・隆起や陥没・崩壊などの変状が発生する。これらが、その場所を利用している人間の生活や各種構造物に対して重大な脅威と損害をもたらす。また、ある時予期せず一気に崩壊してしまう可能性を秘めている。

現在、四国には 1200 箇所以上の地すべり防止区域が存在している。地すべりによる災害を未然に防ぐため、あるいは災害を最小限に食い止めるためには、地すべりの発生メカニズムを解明する必要がある。そして、そのメカニズムの解明のためには、三次元的に地すべりの挙動把握を行なう必要がある。しかし、現在地すべりの変位観測には、孔内傾斜計や伸縮計、GPS などを用いて行われている。これらの計測方法は、点の計測であるため、面的にどの部分がよく動いているのか把握することが困難である。

1-2. デジタル写真測量とレーザースキャナ計測

地すべり地では、地すべりにより滑落崖などが発生し、複雑な起伏をしている。そのため、地すべり地形を表現するには何万もの点の計測が望まれる。高性能なデジタルカメラ（300 万画素以上）を用いれば、何万もの点を取得することは可能である。デジタルカメラを用いた地すべり挙動把握の研究の一つに、野村らの行った「デジタル写真測量による三次元地形モデルの自動生成」[01] がある。この研究により作成されたシステムでは、撮影距離 20m、カメラ位置 B/H 比 1 で撮影した画像を用いれば、6㎝ 未満の精度の三次元地形モデルを期待することができるとされている。しかし、基準点データの作成（基準点観測を除く）から三次元座標取得（2571 点）までに約 7 時間要している。野村らの用いた、デジタルカメラと PC の性能を表 1.1 と表 1.2 に示す。

デジタル写真測量の現地での作業は、基準点計測を省けば対象物の撮影のみである。そのため、現地での観測は一箇所あたり 5 分もあれば完了する。しかし、三次元座標を取得するためには、撮影した画像から複雑なプロセスを経てデータを取得しなければならない。その点、レーザースキャナ計測は一度の計測に 20 分ほど時間を要するが、レーザースキャナを基点とする三次元座標を即座に取得することができる。従って、レーザースキャナ計測はデジタル写真測量に比べはるかに測定能率が高いと言える（表 1.3）。

表 1.1：野村らの用いたデジタルカメラの内部標定要素

焦点距離	8.2979mm	
画像サイズ	u ： 2400pixel	v ： 1800pixel
CCD サイズ	W ： 7.5239mm	H ： 5.6500mm
中心点	3.8071mm	2.9107mm

表 1.2：野村らの用いた PC の性能

CPU	AND Athlon XP 2000MHz
MEMORY	DDR RAM 524MB
OS	Windows2000

表 1.3：写真測量とレーザースキャナ計測の比較（カタログ値）

手法種別	写真測量	レーザースキャナ計測
使用機器例	6008metric	LMS-Z210
測定媒体	写真	近赤外レーザー
データ取得法	写真から間接取得	直接取得
測定制限	暗闇は不可	暗闇でも可
測定精度	1mm 以下～数 10cm	25mm
測定能率（点/1h）	10^1～10^2	10^6

1-3. 市販のソフトウェアについて

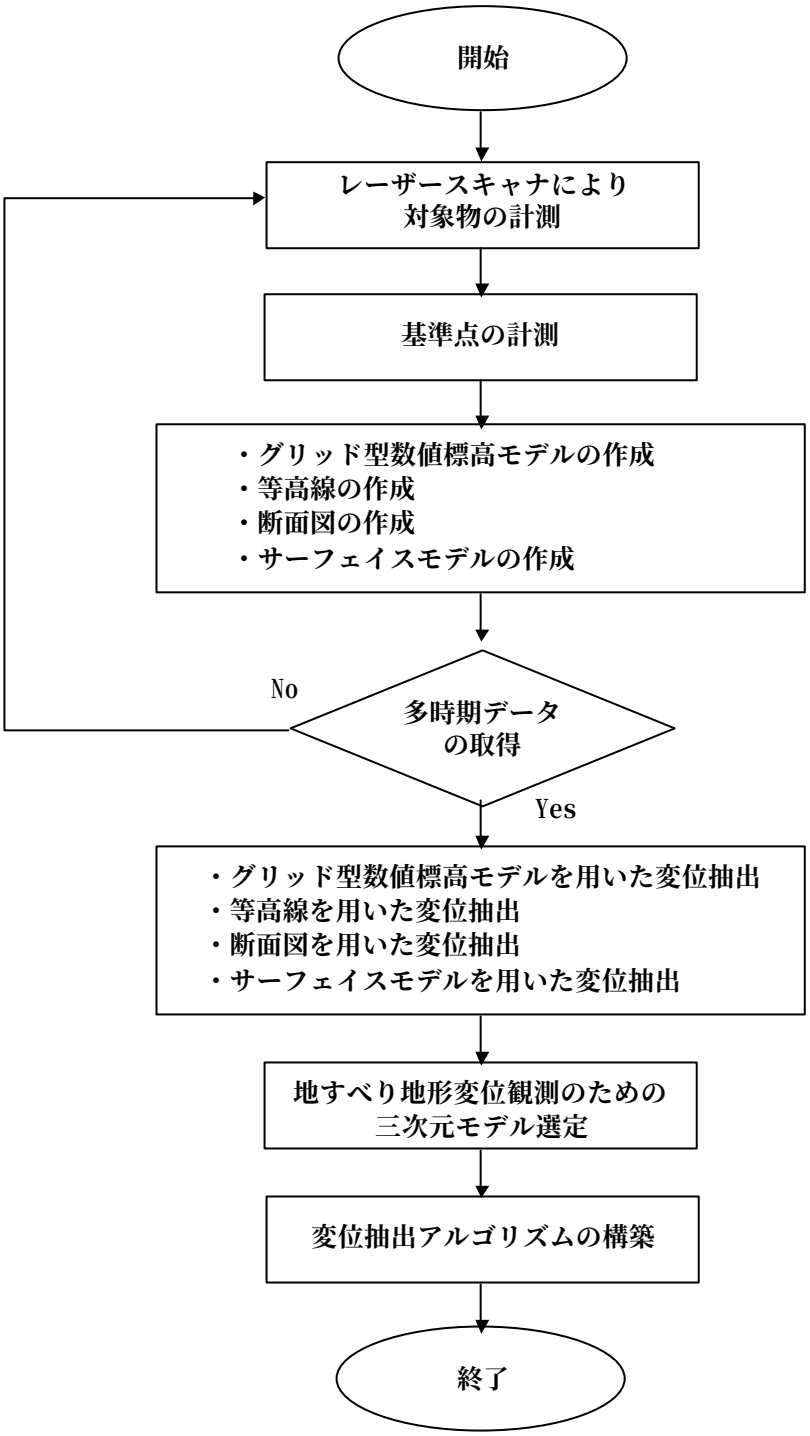
レーザースキャナ計測は、面的な情報を捉えるのに非常に有効的な手段である。しかし、レーザースキャナ取得データは、何百万点にもなるため、そのデータ量は膨大なものとなってしまう。そのため、市販されている CAD ソフトや CG ソフト、GIS ソフトでは、膨大なデータ量であるレーザースキャナデータを開くことが困難である。もし、開くことが可能だとしても、その膨大なデータを扱うことは極めて難しい。また、市販されているソフトでは、計算過程が不明確である。そのため、レーザースキャナデータを扱うには、自作プログラムにより行なうことが必要である。

2. 目的

本研究では、面的な情報を捉えるのに非常に有効的な手段であるレーザースキャナ計測を用い、地すべり地の計測を多時期分行なう。そして、取得したポイントデータより、数値標高モデル、等高線、断面図、サーフェイスモデルを作成し、それぞれのモデルを用いて変位量抽出を行なう。その後、変位抽出に適したモデルを選定し、変位抽出アルゴリズムの構築を行なう。また、対象は高知県仁淀村長者地すべりとする。

以下に研究のフローを示す。

［本研究のフロー］



3. 現地概要

3-1. 位置

長者地すべりは、高知市の西北西約 40㎞ にあり、一級河川仁淀川の支流である長者川右岸に位置している（所在地：高知県高岡郡仁淀村（図 3.1））。

地すべりブロックの規模は、幅約 200m・長さ約 900m・平均斜面勾配約 15°であり、移動方向は北方向(長者川方向)に移動している（図 3.2)。地すべりの末端部は長者川を越えており、対岸の護岸工に隆起現象が認められる。

図 3.1：長者地すべりの位置図

3-2. 地形

地すべり地の周辺は山地に囲まれており、その標高は約 600m である。一部露岩が認められ一般に表土は薄い。山頂部には、二重山稜線(稜線が二重になっている)の地形もあり特異である。

長者川沿いの地すべり末端部付近では、勾配 5°～3°幅 100m 程度の谷底平野と、部分的に現河床より比高 5～10m の段丘面（砂礫や砂が多く過去に河川があった平らな地面）が認められる。長者川の河床は幅 50m 程度で、河床には 3～10m の転石が分布している。

3-3. 地質

地すべり地基盤の地質は秩父帯に属する黒瀬川構造体の伊野層で、泥岩・砂岩・礫岩・花崗岩、石灰岩・蛇紋岩からなる。蛇紋岩は、やわらかい粘土になっている部分があり、すべり面になりやすい特性を持つ。

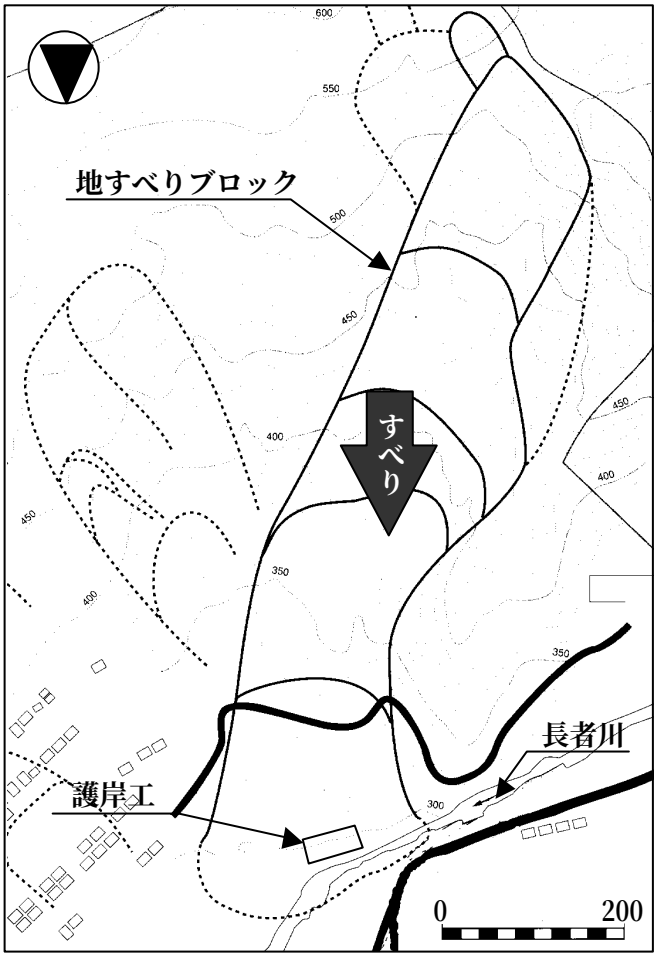


図 3.2：長者地すべりの全景

3-4. 対策工と地すべりの動き

明治 19 年の台風より長者川が大洪水になり移動を始める。その後も、台風や豪雨などにより約 1m/年の移動あり。昭和 26 年より調査が開始され、昭和 28 年より対策工が行われる。しかし、現在も約数 cm～10cm/年の移動が見られる。そのため、変位観測を行ないやすい地域であると考え選定した。

4. レーザースキャナ

4-1. 概要

地上において使用することを目的としたスキャナタイプのレーザーセンサであり、ノンプリズムタイプの光波測距儀の一種である。レーザースキャナは、写真を撮るように、一般的な単点タイプの光波測距儀よりも、高速高密度に位置情報を取得可能である。

4-2. 計測原理

原理は、トータルステーションと類似している。対象物に向かって放射したレーザーパルスが反射して戻ってくるまでの時間により斜距離を計測し、機械を基準とした水平角と垂直角を計測することにより、計測点の三次元座標を求める（図 4.1）。しかし、この時点ではレーザースキャナを中心とした三次元座標である。よって、座標変換を行なうには、基準点（4 点以上）が必要となる。

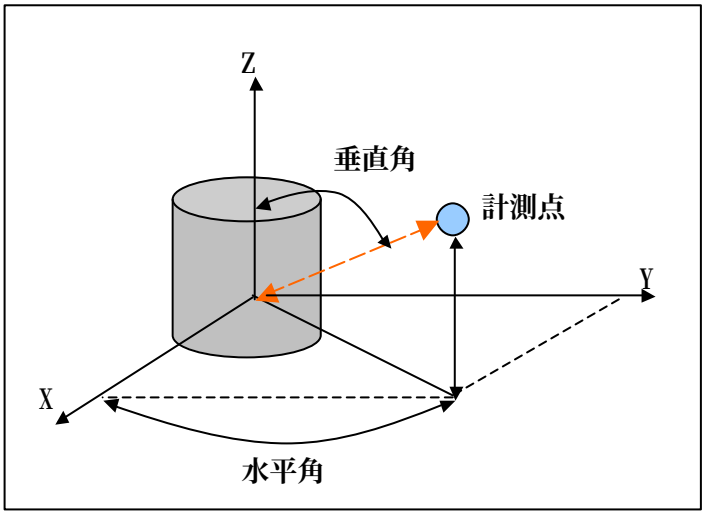


図 4.1：レーザースキャナの計測原理

4-3. 使用したレーザースキャナ（名称：LMS-Z210（Riegl 社製））

現在、レーザースキャナには様々なタイプのものが存在している。地形・地物の計測に有効な「中～長距離計測」タイプ（10～1000m 離れている場合に有効なもの）を表 4.1 に示す。表 4.1 より、測定距離を期待するなら LPM=2K となり、測距精度を期待するなら Cyrax が候補に挙がる。しかし、本研究で最も期待することは、測定能率である。そのため、LMS-Z210 が最も適切であると考え選定した。

LMS-Z210 により得られるデータは、対象物までの距離、角度、対象物の反射強度、カラー情報である。これらのデータは、同梱されているソフト（RiSCAN PRO）により三次元座標への変換も容易に行なえる。また、距離測定器性能、スキャニング性能、1 回の計測で取得できるデータの諸元を表 4.2、表 4.3、表 4.4 に示す。このレーザースキャナは、ミラー

が縦方向に回転しながらデータを取得するため、ライン方向は上から下向きと定義されている（図 4.2）。

表 4.1：中～長距離計測用の地上型レーザースキャナの種類（カタログ値）

機種 （メーカ）	LPM=2K (Riegl)	LMS-Z210 (Riegl)	INT5000 (NEC)	Cyrax (Cyra)
測距範囲（m）	<2500	<350	<1000	<100
最大測定能率（点/秒）	2～3	<9300	0.2	<1000
測距精度（標準偏差）	25mm	25mm	20mm	6mm

表 4.2：LMS-Z210 距離測定器性能（カタログ値）

測定距離範囲	≦350m（反射率≧80%の自然物）
	≦150m（反射率≧10%の自然物）
最短距離	2m
測定精度	±2.5cm
レーザー波長	0.9μm（近赤外線）

表 4.3：LMS-Z210 スキャニング性能（カタログ値）

スキャニング方向	ラインスキャン（縦方向）	フレームスキャン（横方向）
スキャニング数	111～1106	463～4621
スキャニング範囲	±40° ＝ 80° total	0° ～ 333 °
スキャニング機構	回転ポリゴンミラー	回転光学ヘッド
スキャニング速度	5 line/s ～ 52 line/s	1 °/s ～ 15 °/s
角度ステップ幅	0.24 °	0.24 °
角度分解能	0.036 °	0.018 °

表 4.4：1 回の計測で取得できるデータの諸元

諸元	最大測定密度	最小測定密度
最大取得測点数（line×frame）	1106×4621 点	111×463 点
データ容量	約 60MB	約 600KB
所要時間	約 15 分	約 25 秒
レーザー射出角度の間隔	0.072°	0.72°
50m 先での測点間隔	約 6.3cm	約 63cm
測定範囲	80°×333°	

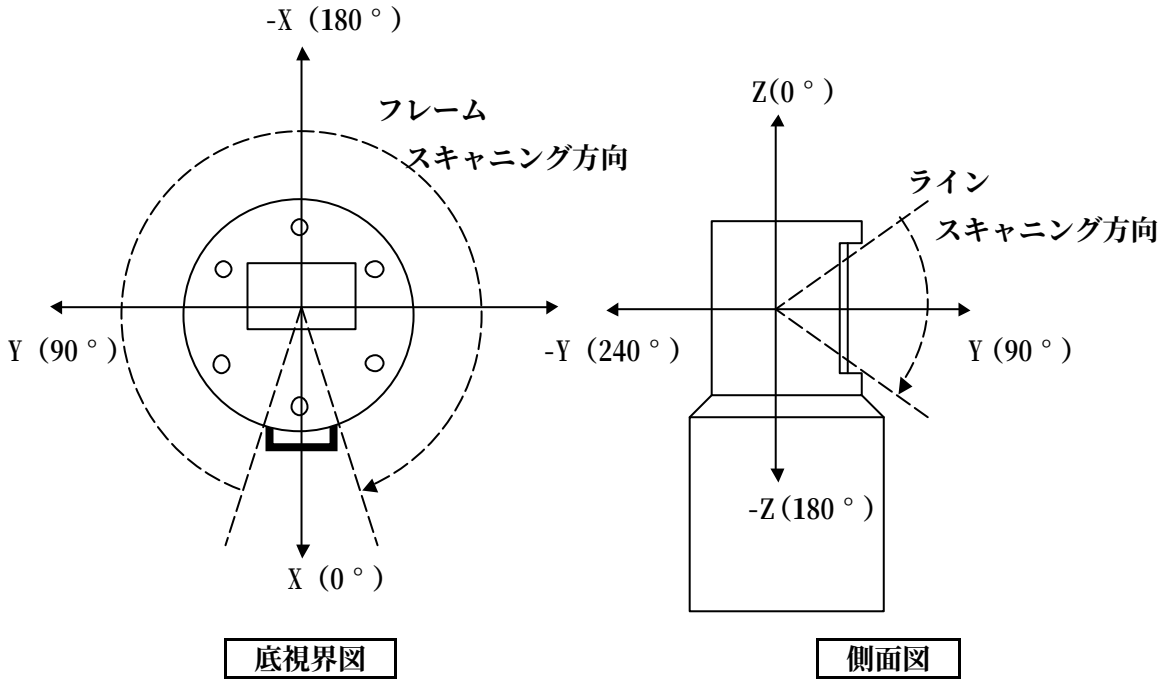


図 4.2：LMS-Z210 のスキャニング方向と軸と角度の定義図

5. データの取得

5-1. 対象・位置

長者川の北側より対岸の護岸工（図 5.1）を計測した。レーザースキャナと護岸工の距離は、近いところで約 45m・遠いところで約 65m 離れている。周りに背の高い植物がなく、見通しがよく護岸工が一望できる場所より計測した。計測した護岸工は、長者地すべりの末端部に位置し、沈降現象なども見られ、最も動きが見られるところである。また、同時期に取得したデータでも計測位置が違う場合、オクルージョンの発生などにより作成する三次元モデルに変化が生じる可能性がある。そのため、計測位置はほぼ同様の位置から行い、傾きや高さもほぼ同様のもので行なった。

地すべりの動きを面的に追跡するためには、地すべり地全体の計測を行なう必要がある。しかし、まずは変位抽出に適した三次元モデルの選定を行ないたいため、最も動きの見られる護岸工の計測を行なった。

図 5.1：計測対象

5-2. 取得データ

変位抽出を行なうため、2 時期分のデータを取得した。1 時期目は 2003 年 7 月 20 日の天候は晴れの日、2 時期目は 2003 年 09 月 18 日の天候は晴れの日である。図 5.2 と図 5.3 は、取得したデータの反射強度画像で、反射強度が 0 の部分を青で、1～255 までを黒～白で表現している。基準点は、両時期十数点ずつ計測しており、その中から両時期共 4 点ずつ選定した（図 5.2、図 5.3）。選定した 4 点は、トータルステーション計測より導いた基準点（絶対座標系、付録 B-01）と、レーザースキャナ計測より導いた基準点（絶対座標系）の差が最も低いものである。この 4 点を用い、RiSCAN PRO により 1 時期目は 1.2cm の精度で、2 時期目は 0.9cm の精度で回転行列の係数を導き出した。2 時期での基準点の位置を変更した理由は、レーザースキャナの機動性を重視したかったためであり、基準点の位置を変更してもほぼ同様の精度でデータ取得が可能であることを確かめたかったためである。

三次元モデル作成に使用するポイントデータは、取得データの護岸工部分のみ（赤線で囲まれた部分）とした。切り取ったサイズは、1 時期目：110×1374（line×frame）、2 時期目：116×1416（line×frame）である。この生データを 3dd2txt.c（付録 B-02）により、TXT 化し、その後 tmtrans.c（付録 B-03）により回転行列の係数を用いて座標変換を行なった。変換前の生データは約 2MB で、変換後の TXT データ約 6MB（約 14.5 万点）である。また、変換後の TXT データは、後の計算が行ないやすいようにスキャニングライン順に上から保存した。

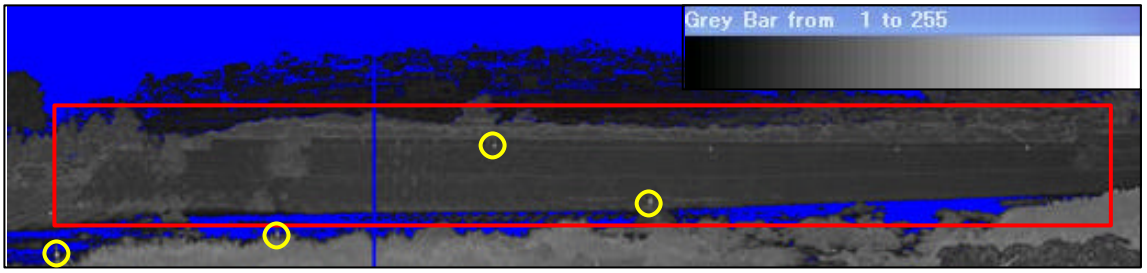


図 5.2：1 時期目の使用データ範囲と基準点位置（反射強度画像）

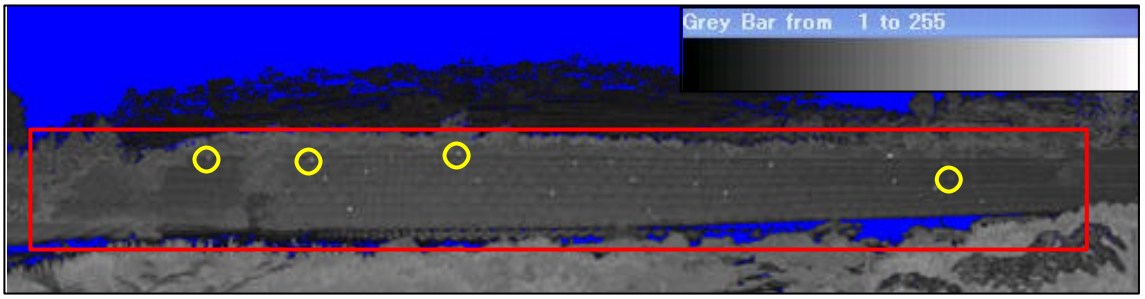


図 5.3：2 時期目の使用データ範囲と基準点位置（反射強度画像）

6. 数値標高モデル

6-1. 数値標高モデルとは

最小単位をピクセル（小さな正方形）とし、そのピクセル毎に地表面の標高値が入力されているものであり、コンピュータで地形を扱う上で欠かせないデータである。しかし、オーバーハング（低い部分より高い部分の方が飛び出している地形）などを表現することはできず、グリッド化する際に値を平均化してしまい空間分解能は下がる。

6-2. 作成アルゴリズムフロー（point2grid.c 付録 B-04）

01）作成する範囲（X,Y）とグリッド間隔を指定し、グリッドサイズを決定する。

02）全てのグリッドに初期値として 0 を入力する。

03）レーザースキャナデータを 01）に当てはめていく。

この時、1 グリッドの大きさによって、複数の点が含まれることがある。その場合は、Z 値が一番低いものを取り作成した。低いものを選んだ理由は、植物の可能性を低くし、標高値である可能性を高くするためである。

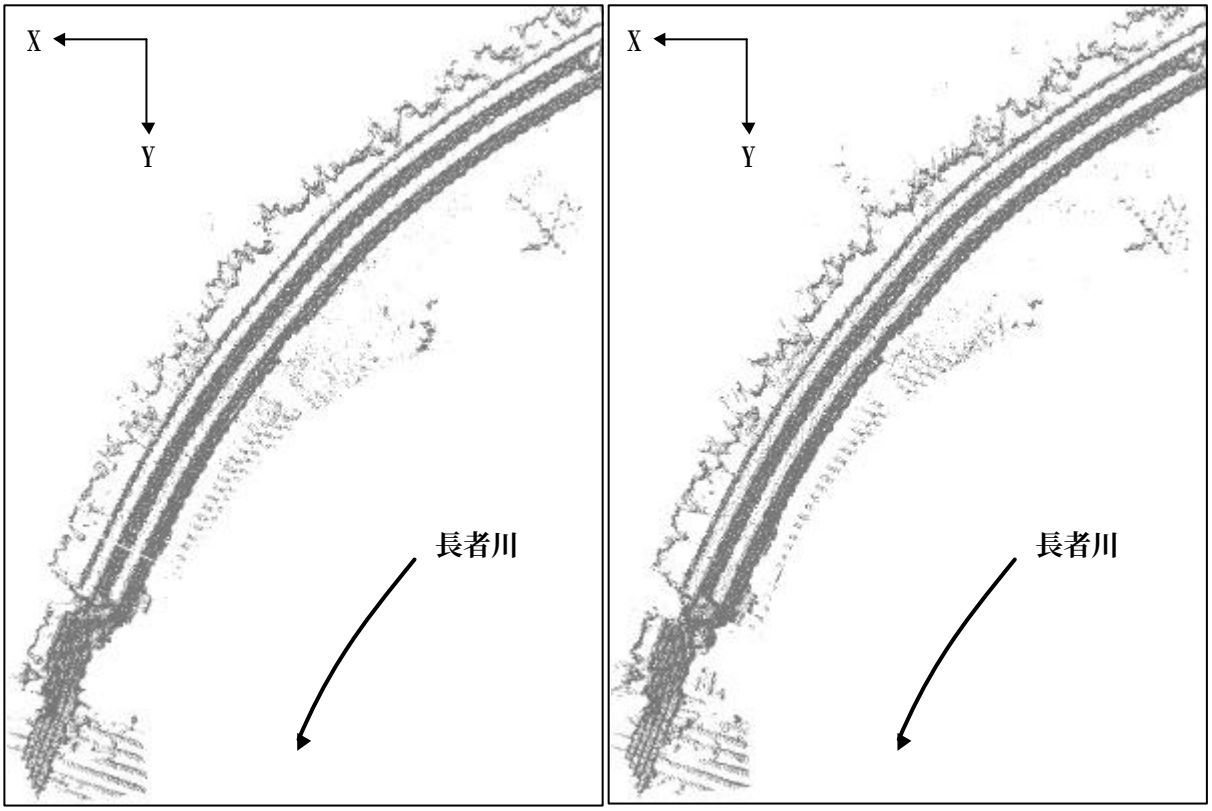


図 6.1：作成した 10㎝ グリッドの数値標高モデル（左：1 時期目、右：2 時期目）

図 6.1 は、10㎝ グリッドでの作成結果である。画像の白い部分は、ポイントデータが 1 点も無かった部分である。作成の結果、天場以外の箇所については、表現することができ

た。データは 2byte で書き込み、ファイル容量は 10㎝ グリッドで約 1Mbyte、1㎝ グリッドでは約 100Mbyte にもなった。1㎝ グリッドで作成した場合でも作成時間はさほどかからなかったが、スキャニング間隔を考えると 1㎝ グリッドでの作成は妥当ではない。今回の取得データは、遠い場所で対象物から約 65m 離れた場所から計測したものであり、最高密度で取得したものである。そのため、スキャニング間隔は約 8㎝ である。よって、No データのグリッドをなくすためには 10㎝ グリッドでの作成が妥当である。

作成したデータは、測定値をそのまま当てはめたものである。よって、基準点計測誤差（約 0.02㎝）＋レーザースキャナの測定精度（約 2.5㎝）＋座標変換の精度（約 1㎝）の約 3.52㎝ の精度で作成可能である。しかし、グリッド化するため値が平均化され、空間分解能が下がる。

6-3. 変位抽出

変位抽出は、画像演算（dem_opt.c 付録 B-08）により行なった。演算箇所は、間違った変位抽出を防ぐため、両時期それぞれが 0 でないグリッドのみ演算した。レーザースキャナでは、固定しない限り同一点の計測は困難である。そのため、作成するグリッド間隔を小さくすればする程、同じグリッドでも両時期共にデータが存在する可能性は減少する。図 6.2 は、変位抽出量の単位を 1㎝ オーダーとし、10㎝ グリッドでの変位抽出結果である。図 6.2 のように変位は全く見られなかった。

今回の対象物では、数値標高モデルを用いた変位の抽出はできなかった。しかし、作成したプログラムでは、グリッド間隔の狭い場合でもモデル作成、変位抽出ともに所要時間はさほどかからなかった。そして、他方向からの統合データでも使用可能である。そのため、地すべり地全体などの変位を大まかに見るためには良いと考える。また、プログラムで読み込む際に、X と Z または Y と Z を入れ替えることにより、YZ 平面または ZX 平面での変位量抽出も可能である。

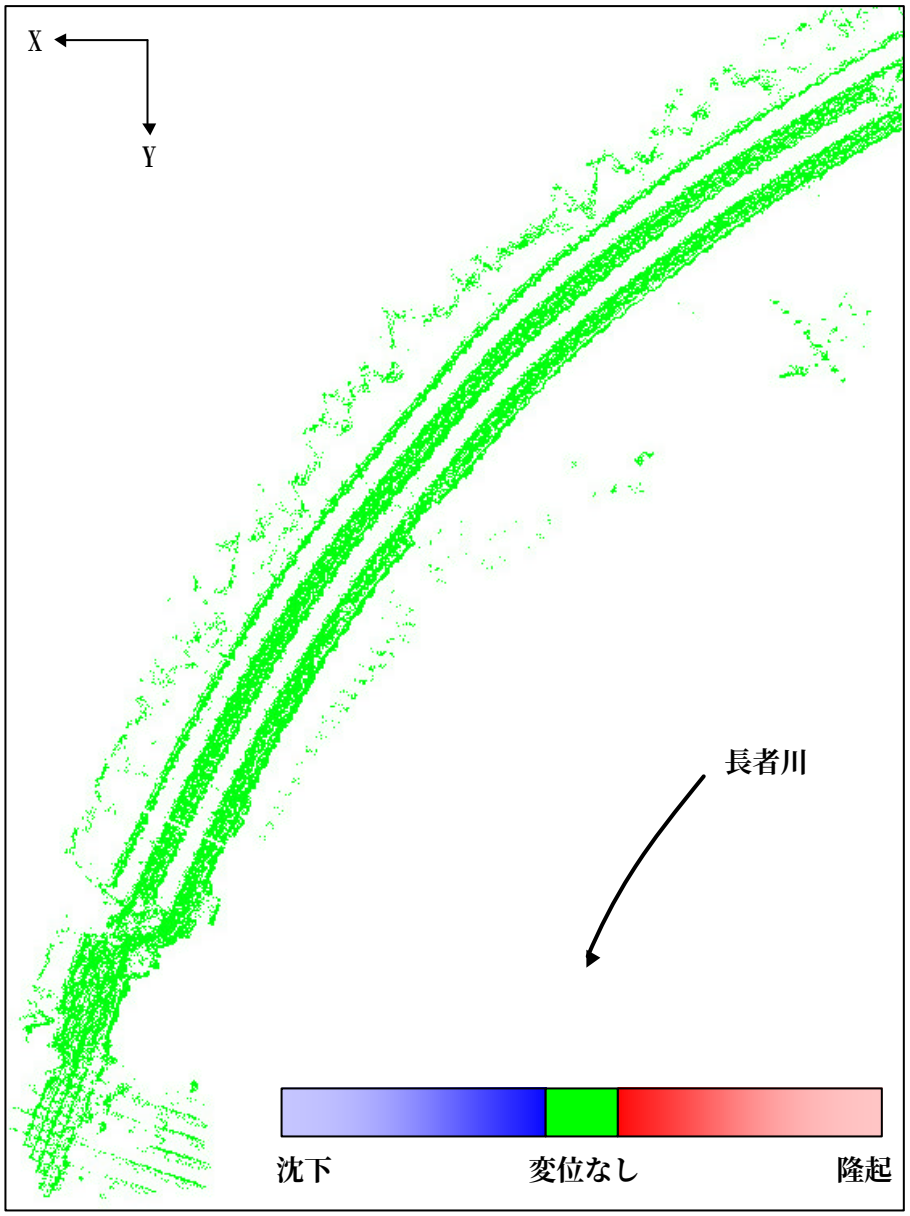


図 6.2：10cm グリッドでの変位抽出結果画像

7. 等高線

7-1. 等高線とは

等高線は、山や谷などの地形の凹凸を表わすために Z 軸に垂直に切断した面を線で表現したものである。等高線の間隔が狭いところでは、傾きが急で、広いところは傾きが緩やかになる。また、等高線作成間隔を狭めることにより、より詳細な表現ができる。しかし、Z 軸に対して平行な面のある地形(崖など)を完全に表現することはできない。

7-2. 作成アルゴリズムフロー（contour.c 付録 B-05）

01）作成したい等高線位置 Z 値を指定する。
02）レーザースキャナデータを 1 ライン毎読み込む。
03）02）の中から 01）に最も近いものを上下それぞれ 1 点ずつ選択する。
04）03）の 2 点間距離が指定範囲内かチェックする。
05）04）の判別が指定範囲内であれば、線形内挿によりプロットする。
　　指定範囲内でない時はプロットせず、線も結ばない。
06）02）に戻る。

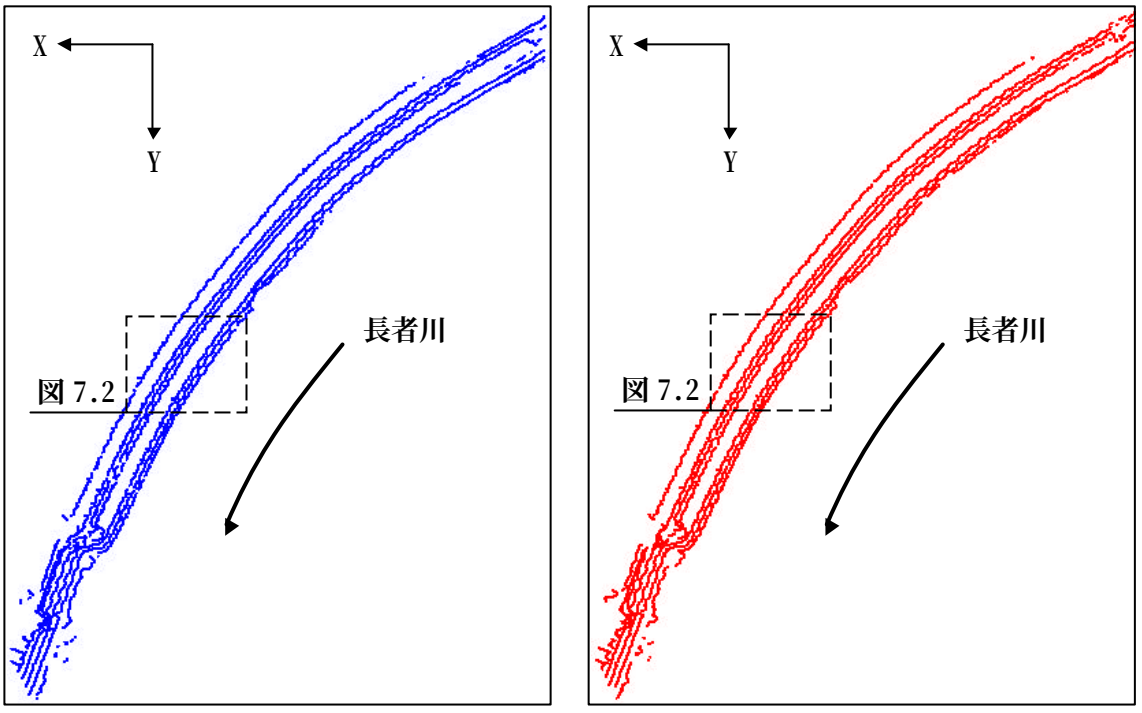


図 7.1：作成した等高線（左：1 時期目、右：2 時期目）

作成の結果、bi-linear 法により作成することができた。図 7.1 は、護岸工ブロックの間隔（約 80㎝）で作成したものである。構成する断面は 7 断面であり、1 断面約 1300 点用いて作成したものである。1 断面約 12Kbyte で全て合わせても約 80Kbyte とデータ量を非常に抑えることができた。また、bi-linear 法により導いているため、基準点計測誤差（約 0.02㎝）＋レーザースキャナの測定精度（約 2.5㎝）＋座標変換の精度（約 1㎝）の約 3.52㎝ の精度で作成可能であると考える。

7-3. 変位抽出

面的に捉えるために、複数断面の変位抽出を行なった。2 時期分のデータを重ねたものを図 7.2 に示す。図 7.2 に示した C1（Z:337.512m）で約 5.5㎝、C2（Z:340.222m）で約 4㎝、C3（Z:342.442m）で約 1.5㎝、変位の大きな場所で約 8㎝ 推定される移動方向とは逆に動きが見られた。導いた変位量は、指定した範囲の等高線と概ね垂直方向の変位を平均したものである。まず、指定範囲の等高線をヘルマート変換により、XY 座標から X 座標に概ね平行になるような uv 座標に変換する。その後、v 軸と平行な線を指定した間隔でひき、1 時期目との交点と 2 時期目との交点の差を導き、平均値を求めた（contour_diff.c 付録 B-09 図 7.3）。護岸工のブロックが沈下している箇所についても調べたが、同様の結果であった。図 7.2 に見られる突起した部分は、護岸工のブロックとブロックの隙間である。ブロックとブロックの隙間については、レーザースキャナの計測条件（位置・高さ・傾き）が少しでも異なると大きな変位となることが考えられる。そのため、今回は無視した。

等高線を用い変位抽出を行なった結果、比較的下段部分のブロックに変位が大きく見られた。さらに、上流部分から下流部分にかけて調べて見たところ、比較的下流部分のブロックに変位が大きく見られた。このように、XY 平面での変位を容易に捉えることができた。

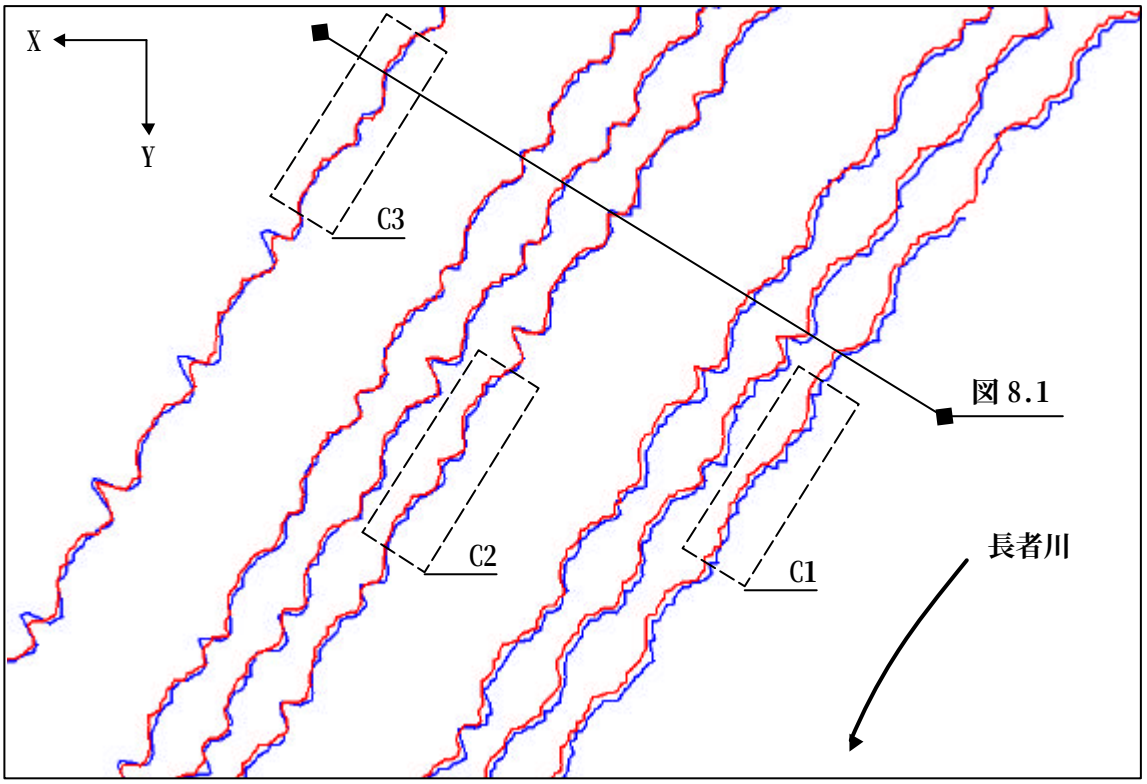


図 7.2：2 時期分の等高線を重ねたもの（青線：1 時期目、赤線：2 時期目）

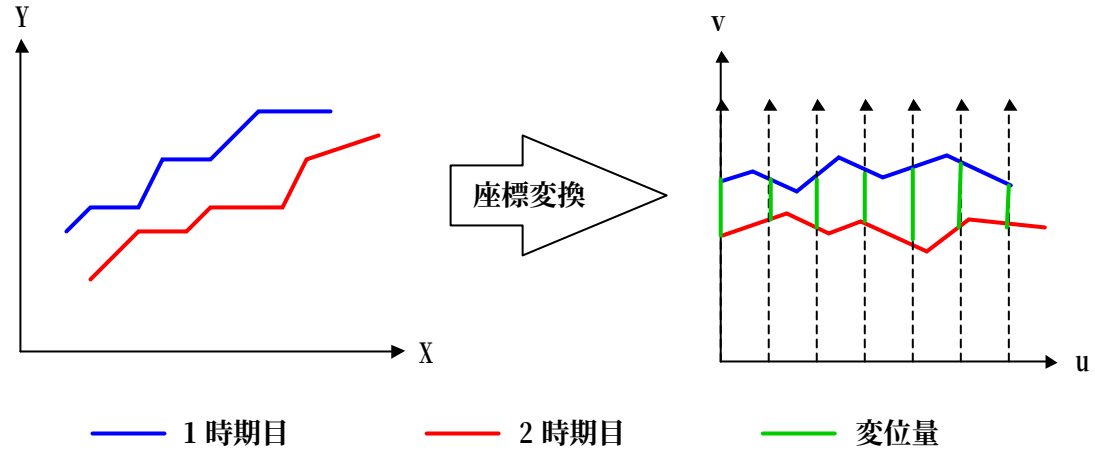


図 7.3：等高線を用いた変位量計算イメージ図

8. 断面図

8-1. 断面図とは

断面図は、Z 軸に平行に切断した面を線で表現したものである。反り返るような崖でも表現することができるため、等高線のみで表現できない地形は、断面図を用いることで表現することができる。

8-2. 作成アルゴリズムフロー（trverse_bi.c 付録 B-06）

01）作成する断面図の始点と終点を指定する。
02）01）より、X-Y 平面上での切断面の直線の式（式 8.1）を決定する。
03）X-Y 平面上で、切断面の式と測点の距離がある一定の範囲内にあるものを抽出する（式 8.2）。
04）03）を Z 値の小さい順に並び変える。
05）04）を上から 2 点毎、座標を読み込む。
06）05）が、切断面をまたいでいるかチェックする。
07）06）が Yes である場合、2 点を結び切断面との交点を導く。
08）線形内挿により切断面上の Z 値を算出する。
09）1 点ずらして 05）に戻る。

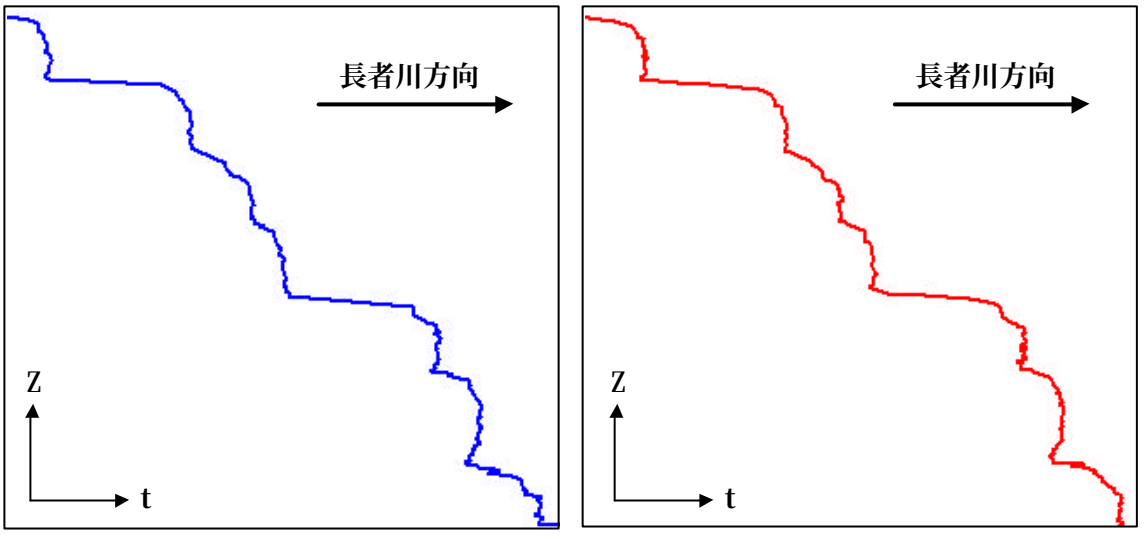


図 8.1：作成した断面図（左：1 時期目、右：2 時期目）

$$x_b = at + x_a$$
$$y_b = bt + y_a$$

x_a, y_a ：始点の座標
x_b, y_b ：終点の座標
a, b ：直線の係数

式 8.1

$$f = (x_i - at + x_a)^2 + (y_i - bt + y_a)^2$$

$$\frac{df}{dt} = 2t(a^2 + b^2) + 2(x_a a - x_i a + y_a b - y_i b) = 0$$

$$t = \frac{-2(x_a a - x_i a + y_a b - y_i b)}{2(a^2 + b^2)} \qquad \text{式 8.2}$$

$$x_t = at + x_a$$

$$y_t = bt + y_a$$

$$d = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2}$$

x_a, y_a	：始点の座標	t	：始点から読込み点までの水平距離
x_i, y_i	：読込んだ点の座標	x_t, y_t	：直線上を始点から t 分進んだところ
a, b	：直線の係数	d	：読込み点から直線までの距離

上に示した作成アルゴリズムフローは、測定値に基づいて導き出す計算手法の bi-linear 法である。最も近い値を導き出す計算手法の nearest neighbor 法についても作成した。両手法により作成した断面図を比較した結果、10^{-4}㎜ 未満の差であった。

断面図は、等高線とは異なり全ての点を線で結んだ。これは、入力するポイントデータを 1 方向のみとしたため、標高値の高いものまたは低いものから結すめば、間違った形になることはないためである。

作成の結果、等高線と同様、bi-linear 法により作成することができた。また、護岸工のオーバーハング（低い部分より高い部分の方が飛び出している箇所）しているブロックも表現することができた。最終的なファイル容量は、図 8.1（約 230 点用いて作成した場合）で、約 2.5Kbyte であった。また、bi-linear 法により導くことができたため、基準点計測誤差（約 0.02㎝）＋レーザースキャナの測定精度（約 2.5㎝）＋座標変換の精度（約 1㎝）の約 3.52㎝ の精度で作成可能であると考える。

8-3. 変位抽出

2 時期分のデータを重ねたものを図 8.2 に示す。等高線と同様、推定される移動方向とは逆方向に変位は見られた。変位量は、等高線と同様の方法により導いた（travers_diff.c 付録 B-10）。図 8.2 に示した T1 で約 6.5㎝、T2 で約 4.5㎝、T3 で約 3㎝、変位の大きな場所で約 8㎝ 推定される移動方向とは逆に動きが見られた。また天場は、1 時期目が下で 2 時期

目が上になっている。この要因としては、レーザースキャナでの計測条件の違いや何らかの力により護岸工が長者川と逆方向に変位したことによる隆起現象の 2 つが考えられるが、どちらの要因かの判別を付けることは困難であった。

断面図を用いた変位抽出を行なった結果、等高線を用いた変位抽出と類似の結果であった。切断方向と Z 軸方向の変位を捉えるためには、良いモデルである。しかし、断面図のみを用いて変位抽出を行なう場合、1 断面のみでは水平方向の変位を捉えることが困難である。そのため、等高線と併用して行なうことが望ましい。

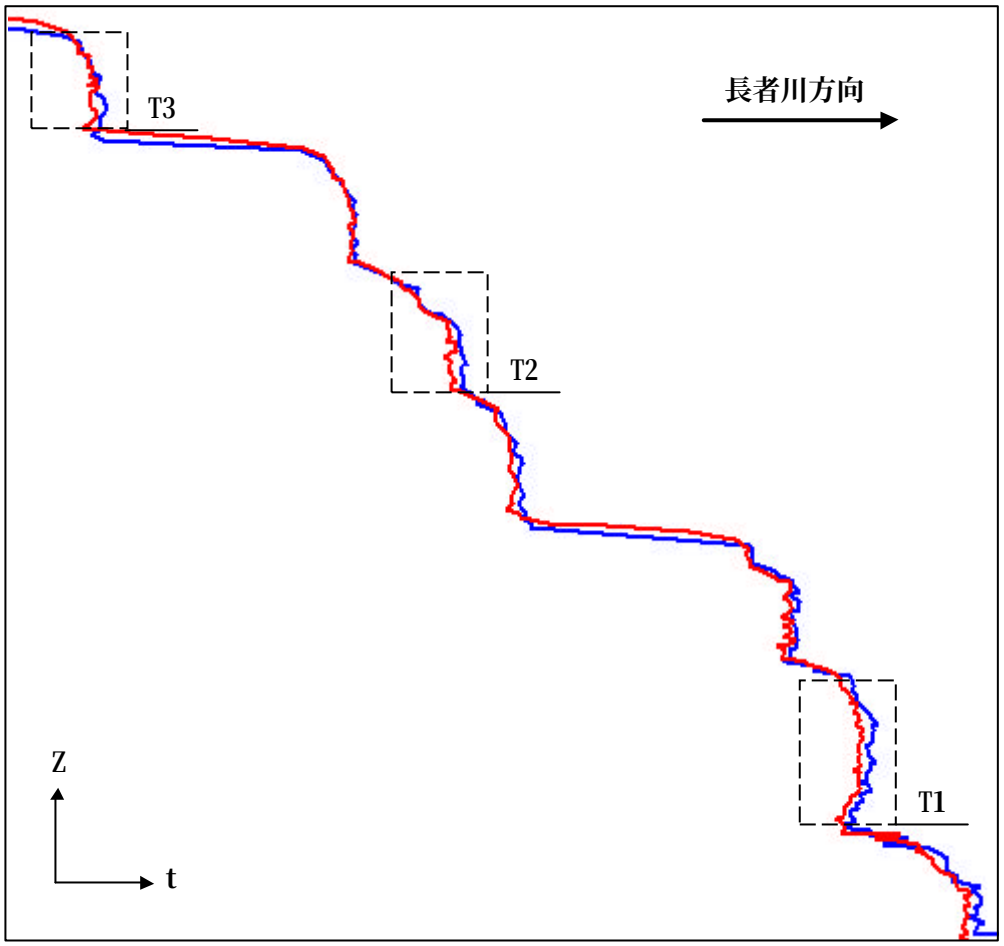


図 8.2：2 時期分の断面図を重ねたもの（青線：1 時期目、赤線：2 時期目）

9. サーフェイスモデル

9-1. サーフェイスモデルとは

多角形の面によって表現したものである。対象物を構成する面毎に色やテクスチャを与えることができるため見かけ上透過せず、ワイヤーフレームモデルなどと比べるとリアルなものを作成することができる。対象物を構成する面の数を増やすことによってリアルさは増す。また、面毎に光の状態を表現するシェーディングと呼ばれる技法を用いることができ、よりリアルな表現が可能である。

9-2. 作成アルゴリズムフロー（txt2dxf.c 付録 B-07）

01）2 ライン毎読み込む。
02）01）から 1 ラインずつ上から 2 点の 4 点を読み込む。
03）02）のうち、3 点以上距離データが存在するかチェックする。
04）03）が 3 点以上存在すれば、レーザースキャナと各 3～4 点との距離が指定範囲以内かチェックする。
05）03)、04）の条件をクリアした 3～4 点を利用してポリゴン化する。
06）縦方向に 1 フレームずらして 02）に戻る。
07）縦方向のポイントデータが無くなれば、横方向に 1 ラインずらして 01）に戻る。

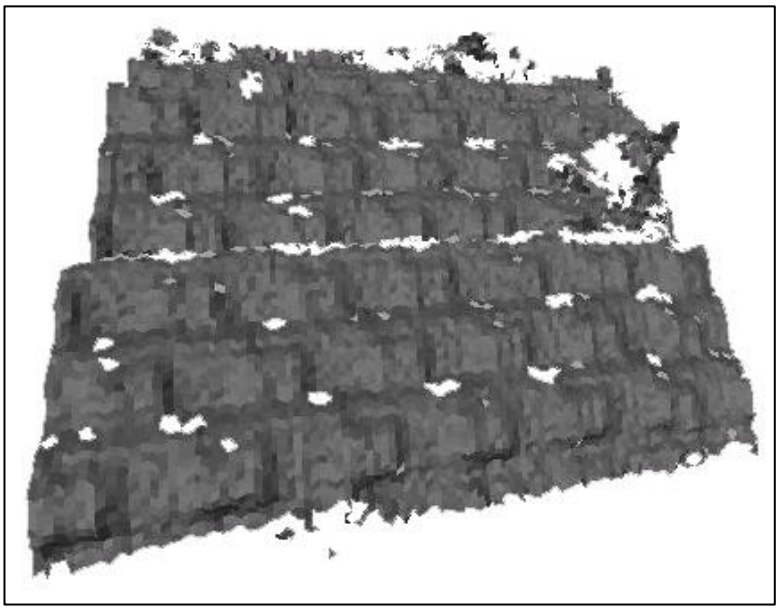


図 9.1：作成したサーフェイスモデル（1 時期目：約 100 ライン）

図 9.1 は、1 時期目の取得データの上流側約 100 ラインをサーフェイスモデル化したものに灰色を付けたものである。作成の結果、護岸工のブロックの形状をリアルに表現することができた。しかし、1 つの面は 3～4 点の 3 次元座標で構成されているため、ファイル容

量は約 25Mbyte（作成サーフェイス＝約 12.5 万面）にもなり市販の CG ソフトでは開くことが困難であった。また、植物のデータを完全に除外することはできなかった。測定値をそのまま用いモデル作成を行なっているため、基準点計測誤差（約 0.02cm）＋レーザースキャナの測定精度（約 2.5cm）＋座標変換の精度（約 1cm）の約 3.52cm の精度で作成可能であると考える。

9-3. 変位抽出

9-3-1. グリッド上でのサーフェイス演算（surface2grid.c 付録 B-11）

サーフェイスモデル同士の演算を行い、結果を変位量として用意したグリッドに入力するというものである（図 9.2)。変位抽出フローを以下に示す。

01）作成する範囲（X,Z）とグリッド間隔を指定し、グリッドサイズを決定する。
02）全てのグリッドに初期値として 0 を入力する。
03）1 グリッド毎読み込み、交点の座標計算（X,Z）を行なう。
04）1 時期目のサーフェイスモデルから 03）を含む面を 2 次元平面で探索する（図 9.3)。判別方法は、直線の式（式 9.1）を用いて、直線より上か下か、右か左かを行い面の中か外かの判別を行なう。
05）04）で 1 面以上存在する場合は、面の式（ 式 9.2）を用いて、それを行列式に変換し、ガウスの消去法により変位抽出方向の座標（Y）を求める。存在しない場合は、09）に進む。
06）2 時期目のサーフェイスモデルから 03）を含む面を 2 次元平面で探索する。
07）06）で 1 面以上存在する場合は、面の式を用いて、それを行列式に変換し、ガウスの消去法により変位抽出方向の座標（Y）を求める。存在しない場合は、09）に進む。
08）対象物により 04）06）で複数の面が存在する可能性がある。そのため、最も低い差を変位量とし、出力する。
09）1 グリッドずらして 03）に戻る。

面は 3～4 点で構成されている。そのためまずは、4 点で構成されている場合はその 4 点を用いて、最小二乗法により式 9.2 の係数 a、b、c を求めた。しかし、4 点で面が構成されている場合、歪みが生じる可能性がある。従って、歪みを考慮するため 4 点で面が構成されている場合は、3 点ずつ用いて係数を求める方法を採用した。

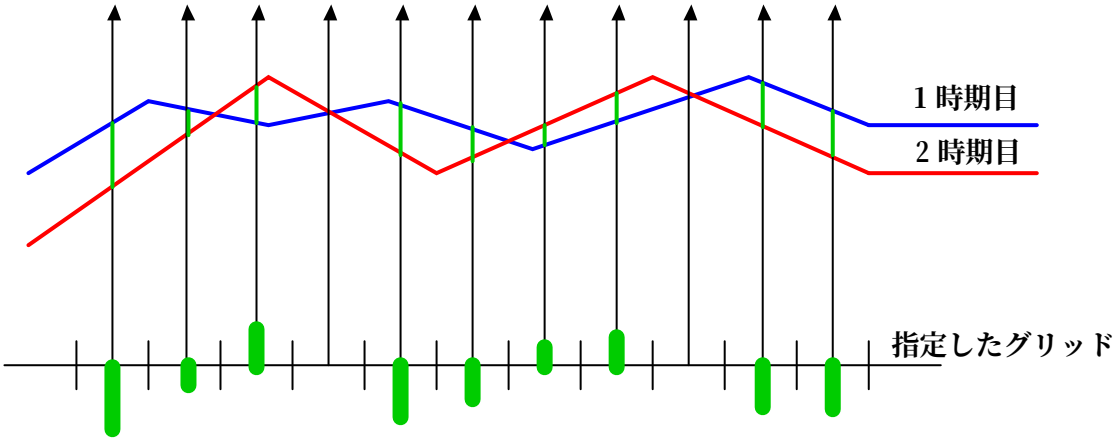


図 9.2：グリッド上でのサーフェイス演算方法のイメージ図（緑線：変位量）

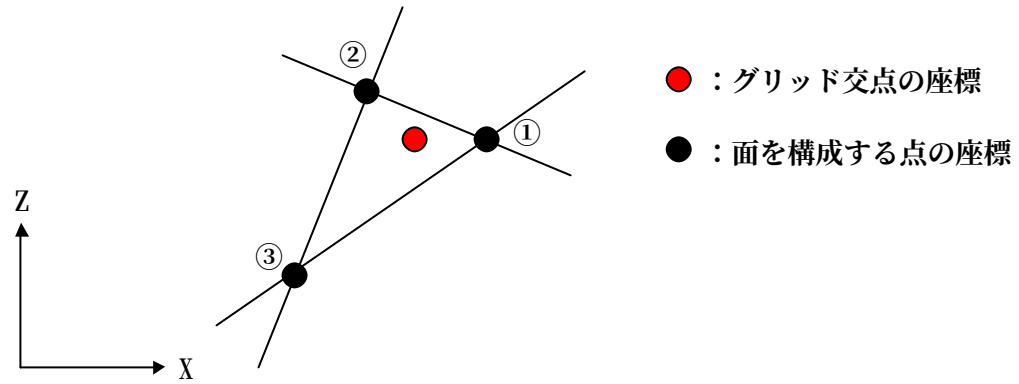


図 9.3：変位算出方向を Y としたときのサーフェイス探索イメージ図

$$ax_i + b = y_i$$

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

式 9.1

x_i, y_i　：直線を構成する点の座標
a, b　：直線の係数

$$ax_i + by_i + cz_i = 1$$

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

式 9.2

x_i, y_i, z_i　：面を構成する点の座標
a, b, c　：面の係数

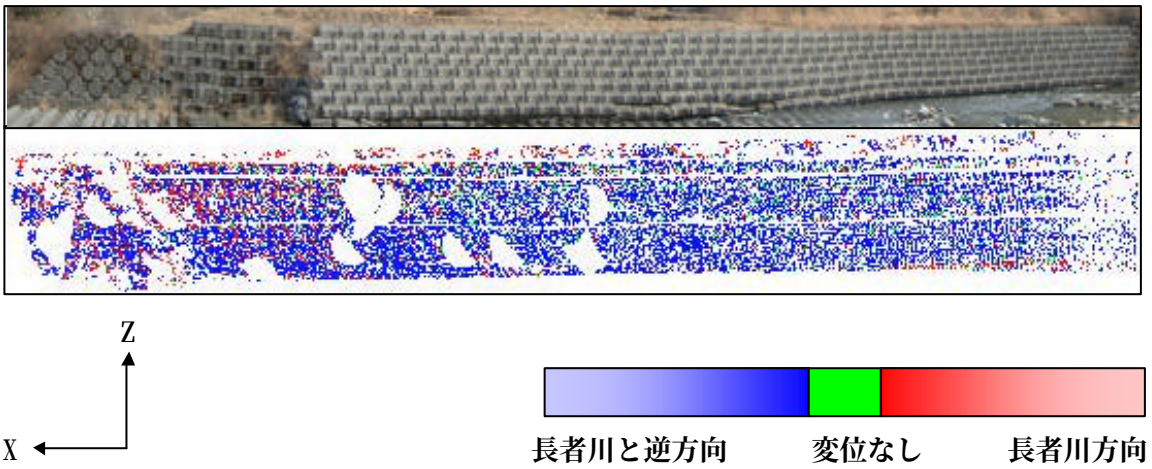


図 9.4：グリッド上でのサーフェイス演算結果（10㎝ グリッド、1㎝ オーダーの変位量）

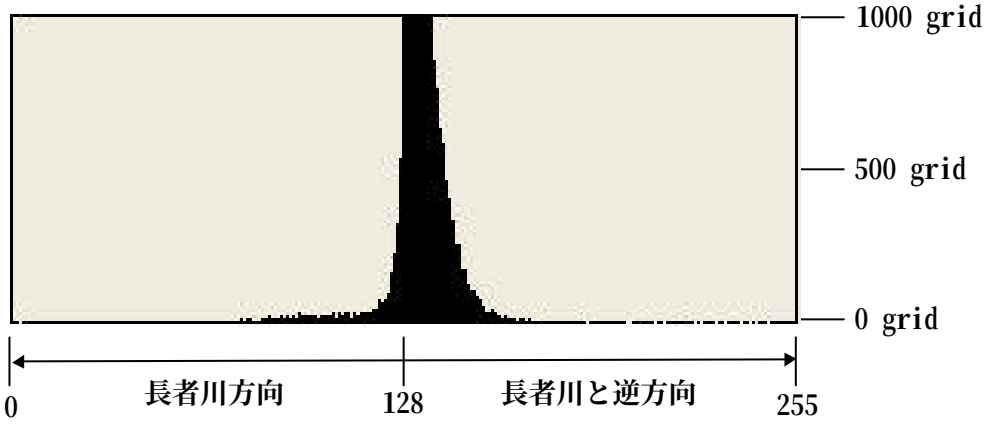


図 9.5：図 9.4 のヒストグラム

図 9.4 は、グリッド上でのサーフェイス演算の結果であり、図 9.5 はそのヒストグラムである。図 9.4 の白い部分は、片方もしくは両時期共にグリッド交点から伸ばした垂線と交わる面が存在しなかった箇所である。変位は、概ね長者川とは逆方向に見られ、約 4㎝ 変位のものが最も多かった（図 9.5）。変位量をグリッドで表現することにより、視覚的にも分かりやすく面的に捉えることができた（図 9.4）。また、XZ 平面でのグリッド同士の演算では、平均化された値同士の演算となるため信頼性は下がるが、サーフェイス同士の演算であるため、測定値に基づいて抽出した変位量である。そのため、精度は下がらないものと考える。しかし、護岸工全体の形状は、曲線を描いているため、面を複数抽出する箇所が存在した。

グリッド上でのサーフェイス演算プログラムは、1 グリッド毎にサーフェイスの探索を行なう。そのため、10㎝ グリッド（65Kbyte）で約 1 時間、5㎝ グリッド（261Kbyte）で約 6 時間も要した。そのため、局所的に行なうには良いが、地すべり地全体の変位抽出には適していないと考える。

9-3-2. 最近隣点探索による変位量ベクトル化（surface2vector.c 付録 B-12）

微小面の重心が両時期で最も近いものを探索し、その点を用いて変位ベクトルを作成した（図 9.6）。作成の結果、三次元的に動きを捉えることができたが、最近隣点探索条件は、“最も近いもの”という条件のみである。そのため、図 9.6 が同一点の移動を捉えている可能性は低いと考える。

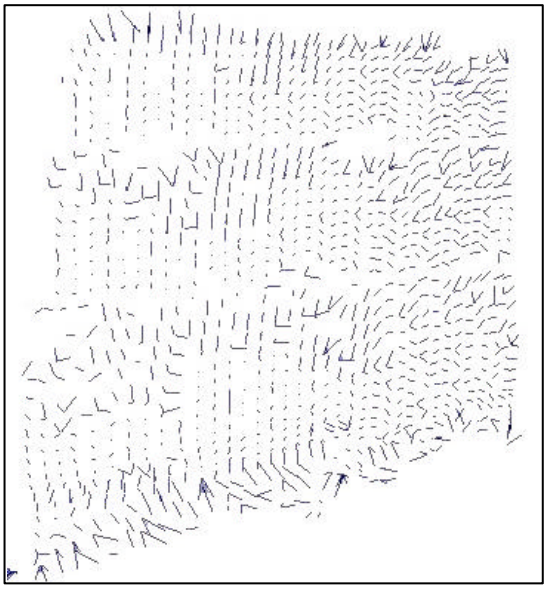


図 9.6：最近隣点探索による変位量ベクトル化（約 100 ラインの下段 3 段部分）

9-3-3. 対象物の自動抽出による変位量ベクトル化（01～05：point2surface.c 付録 B-13）

サーフェイスモデルを用いるのでは無くポイントデータを用い、対象を面として指定した面の自動抽出を行なう。その後、抽出したポイントデータより面の重心を求め、ベクトル化するものである（図 9.7）。変位抽出フローを以下に示す。

01）抽出する面の概ね中心の点を指定する。
02）01）と読み込んだ点間の距離が、指定範囲内のものを指定した点数分抽出する。
03）02）を用い、最小二乗法により面の式の係数を導く。
04）1 点ずつ読み込んでいき、作成した面の式 03）より面から点までの距離を求める。
05）04）が指定範囲内のもののみ抽出する。
06）05）を平均化し、面の重心を導く。
07）2 時期分の 06）を用い、ベクトル化する。
＊抽出する面が複数存在する場合は、以上のプロセスを複数回繰返す必要がある。

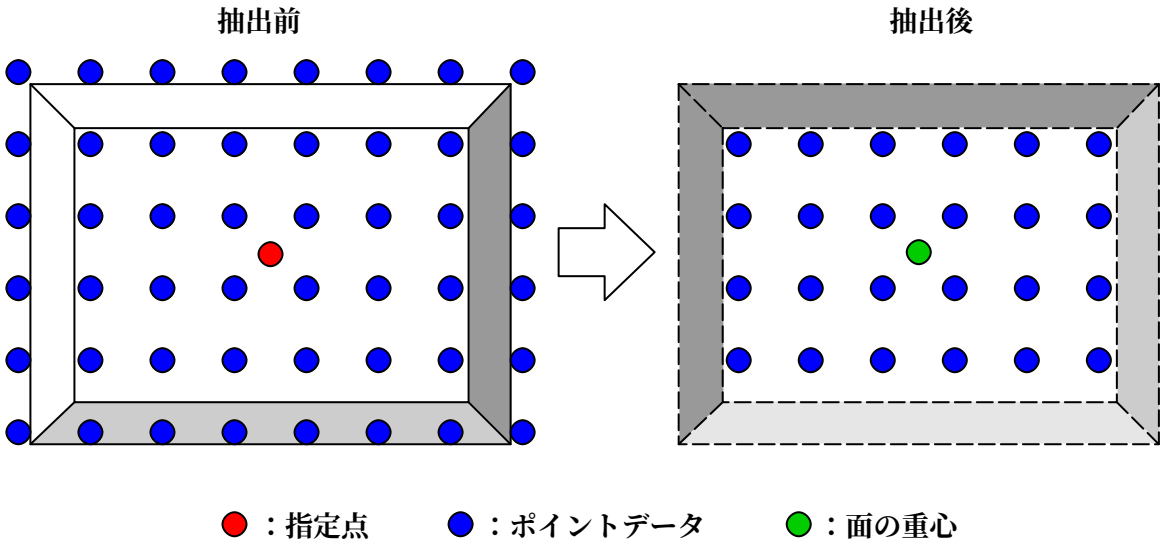


図 9.7：対象物の自動抽出イメージ図

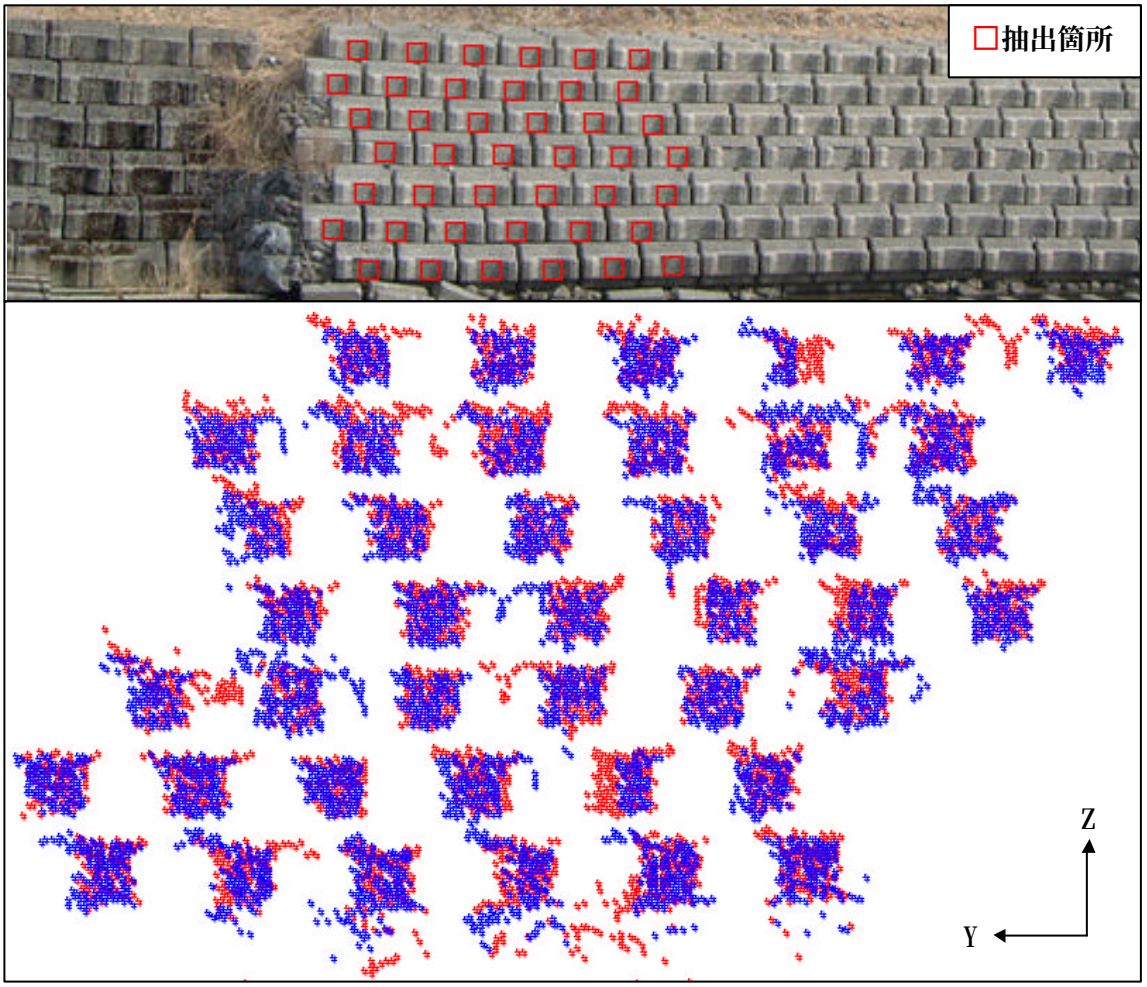


図 9.8：対象物の自動抽出結果（下流側から 42 ブロック分、青：1 時期目、赤：2 時期目）

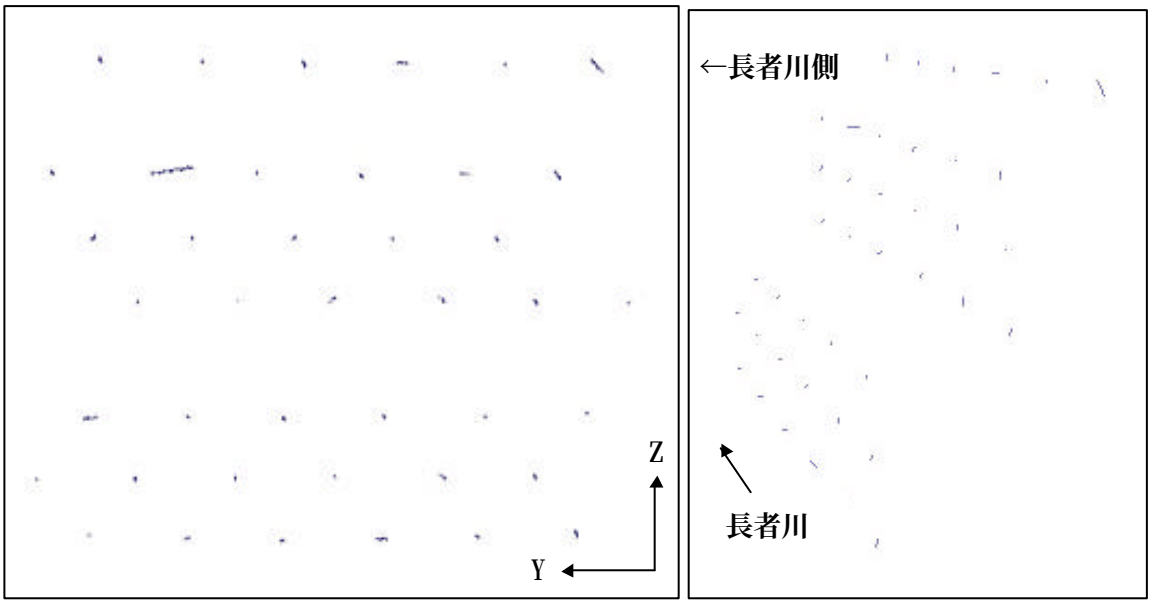


図 9.9：面抽出による変位量ベクトル化（下流側から 42 ブロック分）

図 9.8 は、最小二乗法に用いた測点数と作成した面から点までの距離の条件を同一にして抽出した結果である。面を概ね抽出できた箇所とそうでない箇所が存在した。要因としては、距離に応じて面に存在する点数の違いが挙げられる。図 9.8 の点を用いてベクトル化したものが図 9.9 である。変位の大きなもので約 60㎝、小さなもので約 5㎝、平均で約 13㎝ であった。

今回の抽出面の大きさおよび計測条件では、適応は困難であると考える。しかし、対象物の抽出面が大きく、対象物を近くから計測できるような条件下であれば、最近隣点探索による変位量ベクトル化より信頼性の高い変位ベクトルを作成可能だと考える。

10. 考察

本研究の目的は、地すべり地形の変位抽出に適した三次元地形モデルを選定し、変位抽出アルゴリズムの構築を行なうことであった。等高線、断面図、数値標高モデル、サーフェイスモデルを用いてモデル作成し、変位抽出を行った結果、モデルそれぞれで異なる利点や欠点を持ち合わせていることがわかった。それぞれ以下にまとめる。

[数値標高モデル]

数値標高モデルは、基本的に標高値を用いて Z 方向について可視化したものである。本研究で作成したプログラムは、XY 平面だけではなく、YZ 平面や ZX 平面も可視化することができる。そのため、X 軸方向、Y 軸方向の変位抽出も可能である。しかし、グリッド化するため値が平均化され空間分解能が下がるという欠点を持つ。従って、詳細な変位抽出には適さないが、大まかに変位抽出を行なう際には最適である。

[等高線・断面図]

等高線・断面図は、非常に小さなデータ量でモデル作成が行なえること、bi-linear 法により任意の等高線・断面図を作成可能であること、変位量は多時期データを重ね合わせることで容易に捉えることが可能であることなどの利点が挙げられる。しかし、どちらかのモデルのみでは 3 次元的に変位を捉えることは容易ではない。従って、等高線と断面図 2 つのモデルを併用して変位抽出を行なう必要がある。

[グリッド上でのサーフェイス演算]

グリッド上でのサーフェイス演算では、サーフェイス同士の演算であるため、グリッド同士の演算のように平均化された値を用いるのではなく、測定値を直接用いて変位抽出を行なうことが可能である。しかし、変位抽出には時間を要してしまうこと、グリッド化するため空間分解能が下がること、対象物により 1 グリッドに複数の面を抽出することがあるなどの欠点が挙げられる。従って、グリッドを用いた変位抽出よりも精度は良く、局所的な変位抽出を行ないたい時に適している。

[最近隣点探索による変位量ベクトル化]

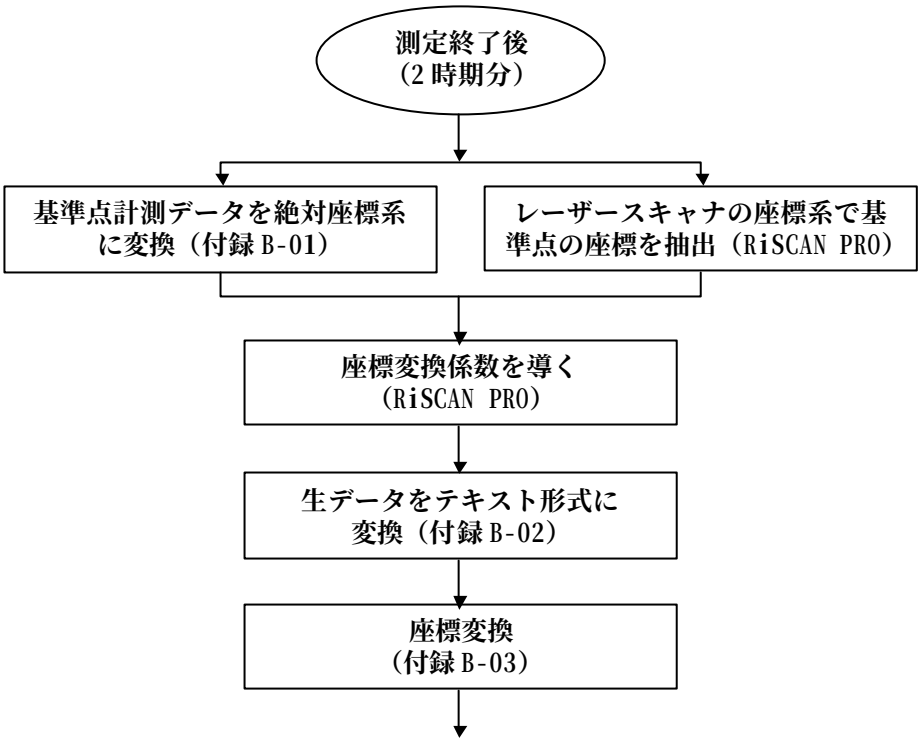
最近隣点探索によるベクトル化は、ベクトル化を行なうことで三次元的に変位を捉えることが可能である。しかし、各微小面の重心に一番近い他時期での微小面の重心を探索した結果であるため、同一点の移動を捉えている可能性は低い。従って、この方法による変位抽出は適用困難であると考える。

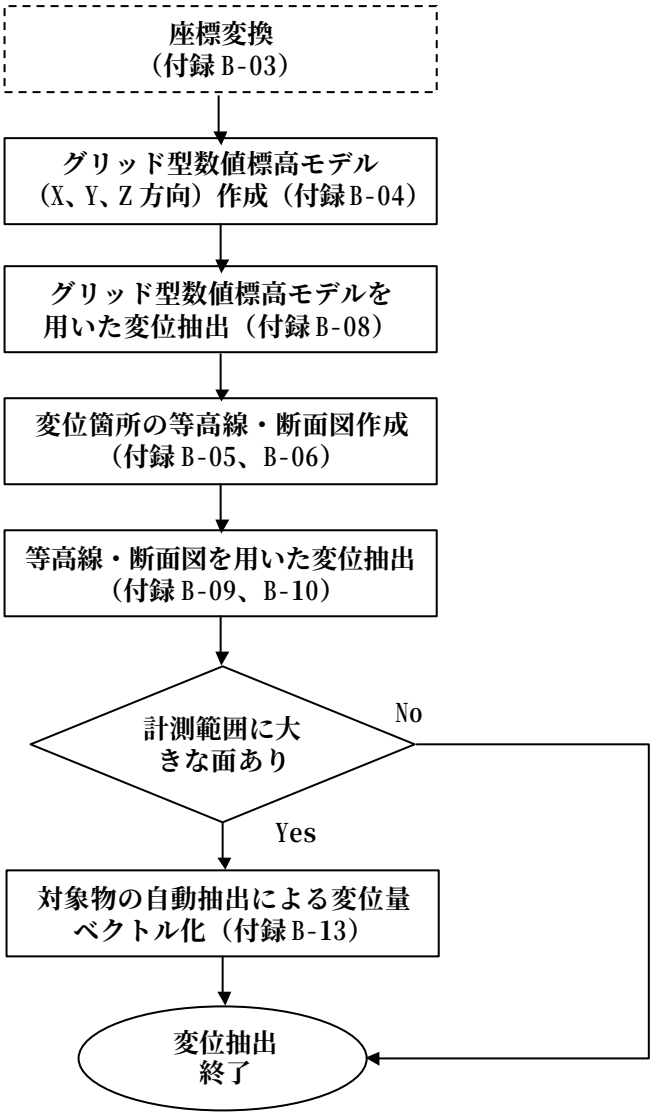
[対象物の自動抽出による変位量ベクトル化]

対象物の自動抽出によるベクトル化は、最近隣点探索によるものと同様に、ベクトル化を行なうことで三次元的に変位を捉えることが可能である。また、この変位抽出方法は、対象とした面のみのポイントデータを抽出し、そのデータより重心を導いた結果である。そのため、最近隣点探索によるベクトル化より信頼性の高い変位ベクトルである。しかし、抽出したい面の概ね中心である点を目視により抽出する必要があること、抽出面の大きさや計測距離により信頼性にバラつきがあることなどの欠点が挙げられる。従って、今回の対象では面積が小さいため適応困難であると考えるが、面がある程度大きく、計測場所より距離が近い対象物であれば適応可能であると考える。

以上より、地すべり地形変位観測には、数値標高モデルを用いた変位抽出、等高線・断面図を用いた変位抽出、ポイントデータを用いた対象物の自動抽出による変位量ベクトル化を使用した変位抽出システムの提案をする。このシステムでは、レーザースキャナの座標系で基準点を抽出する際と座標変換係数を導く際以外は独自のプログラムにより行なうことが可能である。そのため、基準点計測器、LMS-Z210（RiSCAN PRO)、パーソナルコンピュータがあれば地すべり地形変位観測を実現可能である。

［変位抽出システム］





本研究で作成した対象物の自動抽出による変位量ベクトル化は、面を対象としたものであるが、この対象を立体にすることでより精度の良い対象物の移動ベクトルを作成することが可能であると考える。しかし、対象をある立体としたときに、レーザースキャナの微妙な位置・傾き・高さの違いにより、2時期間でオクルージョンが発生する場合と発生しない場合があると考えられる。今回の計測対象で言えば、1時期目より2時期目の方が設置したレーザースキャナの高さが1m高かったとすると、護岸工の天場部分が2時期目では取得できたが1時期目では取得できなかったという部分が存在する可能性が高い。このように、立体の一部分が取得できたものと取得できなかったものでは、比較困難となる。単に対象物を面から立体に変えるだけでも、様々な問題が発生すると考えられる。

また今回の対象は護岸工としたため、植物の測点が非常に少なく済んだ。しかし、地すべり地は大半が植物で覆われている。そのため、植物のデータをどのように扱うか今後の課題となる。

11. 参考文献

01）野村努、「デジタル写真測量による三次元地形モデルの自動生成」、高知工科大学大学院 2002 年度修士論文

02）野村努・高木方隆、「デジタル写真測量による地すべり地の三次元移動追跡への適用可能性」、［287-290P］、日本写真測量学会平成 12 年度秋季学術講演会発表論文集 2000

03）村井俊治、「空間情報工学」、社団法人日本測量協会

04）動体計測研究会、「イメージセンシング」、社団法人日本測量協会

05）高知県土木部防災砂防課、「長者地すべり」

06）リーグルジャパン株式会社、「LMS-Z210 取扱説明書」

07）小田三千夫・織茂郁・松田健也、「航空機レーザースキャナによる数値標高データの取得」、[3-10P]、空間情報技術の実際、社団法人日本測量協会

08）赤松幸生・天野正博・今井靖晃・勝木俊雄・瀬戸島政博・高橋正義・福田未来・船橋学、「森林域における航空機レーザースキャナの利用に関する検証」、[11-16P]、空間情報技術の実際、社団法人日本測量協会

09）中村三友、「航空機レーザースキャナ計測データからの等高線作成処理」、[17-22P]、空間情報技術の実際、社団法人日本測量協会

10）小野尚哉、「地上型 3D レーザースキャナによる災害地の計測」、[111-119P]、空間情報技術の実際、社団法人日本測量協会

11）本郷賢兒、「地上型 3D レーザースキャナによる文化財計測」、[120-126P]、空間情報技術の実際、社団法人日本測量協会

12. おわりに

本研究のはじまりは、レーザースキャナの有用性を期待し購入したことに始まる。前年度までは、デジタル写真測量を用い三次元モデルの生成を行なっていた。計測自体は、即座に終わり、コストが安いということから使用していたが、三次元モデル生成までには大変な苦労を感じた。そのため、レーザースキャナにかける期待は人一倍であった。しかし、購入後実際に使用したところ、確かに期待していた通り計測物の三次元データを即座に取得することはできたが、三次元データと同時に取得するカラー情報が非常に不鮮明であることや、データ量が膨大になるなどの欠点が見付かった。データ量が膨大であるため、市販のソフトでの読み書きが困難であった。レーザースキャナデータを読み込ますために、何時間も待ったこともある。このように、レーザースキャナの利点である「数百万点ものポイントデータを即座に取得可能である」という箱を開けてみると、「データ量が膨大過ぎて市販のソフトでは対応困難である」という欠点であった。レーザースキャナデータを読み込むために購入した市販のソフトもお蔵行きというわけである。「1000 万円以上するのにこれか・・・」と何度思ったことでしょう。

今思えば、私の苦悩はレーザースキャナが来たときからはじまっていたのかもしれません。高価過ぎるため、ヒヤヒヤしながらセッテングし、市販のソフトでは対応困難であり、その後の三次元モデル作成、変位抽出にも頭を悩ませました。慣れない作業（プログラム作成）で、頭がウニになることは度々でした。しかし、プログラムが完成したときは、非常にうれしく達成感を非常に感じることができましたし、研究室のみんなのおかげで楽しく研究を進めることができました。また、こういった機会を与えて頂いたおかげで思考力も付いたように思います。

13. 謝辞

本研究の機会を与えて下さり、数々の有益なご指導を与えて下さいました社会システム工学科の高木方隆助教授に深甚の謝意を表します。

副指導教官としてご助言して下さいました知能機械システム工学科の竹田史章教授に心から感謝致します。

様々な発表の場において適切なご助言をして下さいました社会システム工学科の島弘教授、藤澤伸光教授、村上雅博教授、那須清吾教授、渡邊法美教授、中田愼介教授に深く感謝致します。

日頃から、研究および英語などのご助言をして下さいました Jong Hyeok JEONG さん、誠にありがとうございました。

貴重な時間を割いて観測に協力してくれた菊池有起君、氏家康二君、徳平早映さん、近藤映宏君、坂井知也君、宮田剛君、地紙洋平君、吉良知佐さん、誠にありがとうございました。特に、同じ計測機器を用いた研究であったため共通点が多く、何度も計測に協力してくれた近藤映宏君に感謝します。

また、日々の生活を共にした社会システム工学科の皆様、いつも我々学生達を支えて下さった西村有加秘書、永森美和秘書どうもありがとうございました。

最後に、影ながら支えてくれた家族と友人に感謝致します。

14. 付録

A. 使用 PC スペック

表 A-1：本研究で用いた PC の性能

CPU	Mobile Intel(R) Pentium(R) 4-M 1.70GHz
MEMORY	1.00GB
OS	Microsoft Windows XP

B. 作成プログラム

B-01. 基準点計測データを座標変換（ts2gcp.c）

```
/*-- conbert for TS data to GCP -----------------------------
  TS 取得データを絶対座標に変換する。
  ＊基準点計測の際、ポールを用いて計測した場合は、
    出力値(Z)よりポール高を引く必要がある。
  <入力 TS データフォーマット>
  TS_P [gXa] [gYa] [gZa]
  PR_P [gXb] [gYb] [gZb]
  TS_h [h]
  No.1 [HAR] [ZA] [H(r)]
  No.2   ・   ・   ・
   ・    ・   ・   ・
  <使用ﾃﾞｰﾀ>
  TS_P : ﾄｰﾀﾙｽﾃｰｼｮﾝ位置の絶対座標(m)
  PR_P : ﾌﾟﾘｽﾞﾑ位置 (Oset) の絶対座標(m)
  TS_H : ﾄｰﾀﾙｽﾃｰｼｮﾝの器械高(m)
  No.* : 計測点取得ﾃﾞｰﾀ
         水平角 HAR(°),天頂角 ZA(°),水平距離 H(m)
  <ﾍﾙﾏｰﾄ変換式>
  Lp =  r*cos(a)
  Mp = -r*sin(a)
  Xp = Lp*cos(θ) - Mp*sin(θ) + Xa
  Yp = Lp*sin(θ) + Mp*cos(θ) + Ya
---------------------- October 05. 2003 maked by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define CHR_SIZE 1024

main (argc, argv)
int argc;
char *argv[];
{
        FILE            *fpi;
        int             i, n;
        char            ibuf[CHR_SIZE], ina[10];
        double          gx[2], gy[2], gz[2], ts_h;
        double          sx, sy, isx, isy;
        double          pi, sit, a;
        double          har, za, r;
        double          lp, mp, np;
        double          xp, yp, zp;

        if(argc != 2) {
                printf("--- Conbert for TS Data to GCP Pro ---\n");
                printf("Usage: %s [fpi.txt] > fpo.csv\n", argv[0]);
                exit(1);
        }

        if ((fpi = fopen (argv[1], "r")) == NULL) {
                fprintf(stderr,"Can not open %s. \n", argv[1]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi);

        /*π ｼｭﾃﾙﾏｰの公式 1896'*/
        pi = 4.0*(44.0*atan(1.0/57.0)+7.0*atan(1.0/239.0)-12.0*atan(1.0/682.0)+24.0*atan(1.0/12943.0));

        i=0;        n=1;
        fprintf(stdout, "name,X,Y,Z\n");
        while (fgets(ibuf, CHR_SIZE, fpi) != NULL) {
                if(i < 2){
                        /*TS_P & PR_P*/
                        sscanf(ibuf, "%s %lf %lf %lf", &ina, &gx[i], &gy[i], &gz[i]);
                        fprintf(stderr, "%s\t%.3lf\t%.3lf\t%.3lf\n", ina, gx[i], gy[i], gz[i]);
                }
                if(i == 2){
                        /*TS_H*/
                        sscanf(ibuf, "%s %lf", &ina, &ts_h);
                        fprintf(stderr, "%s\t%.3lf\n", ina, ts_h);
```

```
                        /*θ*/
                        sx = gx[0] - gx[1];
                        sy = gy[0] - gy[1];
                        isx = sx < 0 ? -sx : sx;
                        isy = sy < 0 ? -sy : sy;
                        sit = sx < 0 ? atan(isy/isx) : pi - atan(isy/isx);
                        //fprintf(stderr, "pi : %.15lf\n", pi);
                }
                if(i >= 3){
                        /*TS Data*/
                        sscanf(ibuf, "%s %lf %lf %lf", &ina, &har, &za, &r);
                        har = (har * pi) / 180;
                        za = ((90-za) * pi) / 180;
                        a = sy > 0 ? -har : har;
                        lp = r * cos(a);
                        mp = -r * sin(a);
                        np = r * tan(za);
                        xp = lp*cos(sit) - mp*sin(sit) + gx[0];
                        yp = lp*sin(sit) + mp*cos(sit) + gy[0];
                        zp = ts_h + np + gz[0];
                        fprintf(stdout, "%d,%.4lf,%.4lf,%.4lf\n", n, xp, yp, zp);
                        n++;
                }
                i++;
        }
        fprintf(stderr, "n : %d\n", n-1);

        fclose(fpi);
        exit(0);
}
```

B-02. 生データをテキストデータ変換（3dd2txt.c）

```
/*-- convert for 3dd to txt ----------------------------------------
  ﾚｰｻﾞｰｽｷｬﾅﾃﾞｰﾀ(***.3dd)をﾃｷｽﾄﾃﾞｰﾀ(***.txt)に変換するﾌﾟﾛｸﾞﾗﾑ。
  1[gon] = 0.9[deg]
---------------------------------- july 25. 2003 made by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define    CHR_SIZE  1024
#define    PIX_SIZE  1106

main (argc, argv)
int argc;
char *argv[];
{
          FILE                *fpi;
          unsigned char       rr, gg, bb;
          unsigned char       ibuf[CHR_SIZE];
          unsigned char       oc1, oc2;
          int                 i, j;
          int                 h_size, d_size, n_size, l_size;
          int                 idat, di;
          double              va, ha;
          double              pi, hd;
          double              x, y, z, d;

          if(argc != 2) {
                    printf("--- Convert for 3DD to TXT Pro ---\n");
                    printf("Usage: %s [fpi.3dd] > fpo.txt\n", argv[0]);
                    exit(1);
          }

          h_size = 89;                    /*ﾍｯﾀﾞ ｻｲｽﾞ*/
          n_size = 6;                               /*n_size * d_size = 1 point のﾃﾞｰﾀ*/
          d_size = 2;

          if ((fpi = fopen (argv[1], "rb")) == NULL) {
                    fprintf(stderr,"Can not open %s. \n", argv[1]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi);

          fread(ibuf, h_size, 1, fpi);
          oc1 = ibuf[13];
          oc2 = ibuf[12];
          l_size = (int) oc1 * 256 + (int) oc2;
          fprintf(stderr, "line : %d \n", l_size);

          //fprintf(stdout, "Frame\tLine\t");
          fprintf(stdout, "X\tY\tZ");
          fprintf(stdout, "\tdi");
          fprintf(stdout, "\tR\tG\tB");
          fprintf(stdout, "\n");
          i = 0; j = 0;
          while (fread(ibuf, d_size * n_size, 1, fpi) != 0) {
                    /* 斜距離 d[m] = idat * 8[mm] / 1000 */
                    oc1 = ibuf[1];
                    oc2 = ibuf[0];
                    idat = (int) oc1 * 256 + (int) oc2;
                    d =  idat * 8.0 / 1000.0;

                    /* 鉛直角 va[deg] = 2 * (idat%13333) * 0.01 * 0.9 */
                    oc1 = ibuf[3];
                    oc2 = ibuf[2];
                    idat = (int) oc1 * 256 + (int) oc2;
                    va =  2.0 * (idat % 13333) * 0.01 * 0.9;
                    //fprintf(stdout, "%10.3lf\t", va/0.9);
                    /*π ｼｭﾃﾙﾏｰの公式 1896'*/

          pi=4.0*(44.0*atan(1.0/57.0)+7.0*atan(1.0/239.0)-12.0*atan(1.0/682.0)+24.0*atan(1.0/12943.0));
```

```
                z = d * sin((90.0-va) * (pi/180.0));
                hd = d * cos((90.0-va) * (pi/180.0));

                /* 水平角 ha[deg] = idat * 0.01 * 0.9 */
                oc1 = ibuf[5];
                oc2 = ibuf[4];
                idat = (int) oc1 * 256 + (int) oc2;
                ha =  idat * 0.01 * 0.9;
                //fprintf(stdout, "%10.3lf\t", ha/0.9);
                x = hd * cos(ha*pi/180.0);
                y = hd * sin(ha*pi/180.0);

                /* 反射強度 di */
                di = (int) ibuf[7];

                /* RGB */
                rr =  ibuf[9];
                gg =  ibuf[10];
                bb =  ibuf[11];

                if(di>=0.0){
                        //fprintf(stdout, "%d\t%d\t", j, i);
                        fprintf(stdout, "%.3lf\t%.3lf\t%.3lf", x, y, z);
                        fprintf(stdout, "\t%d", di);
                        fprintf(stdout, "\t%d\t%d\t%d", rr, gg, bb);
                        fprintf(stdout, "\n");
                }

                i++;
                if (i==l_size) {
                        i = 0;
                        j ++;
                        if (fread(ibuf, 10, 1, fpi) == 0) {
                                break;
                        }
                }
        }
        fprintf(stderr, "frame : %d \n", j);

        fclose (fpi);
}
```

B-03. 回転行列係数を用いた座標変換（tmtrans.c）

```
/*-- coordinate transformation using SOP file --------------------
  3D-RiSCAN より抽出した SOP(回転行列)を用いて、座標変換する。
------------------------------ August 07. 2003 made by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define CHR_SIZE 1024
#define SOP_SIZE 20

main (argc, argv)
int argc;
char *argv[];
{
        FILE            *fpis, *fpit;
        int             i, l, p, rp;
        int             di, r, g, b;
        char            ibuf[CHR_SIZE];
        char            sb[3], in[SOP_SIZE];
        double          c[3][3], rxyz[3];
        double          txyz[3], xyz[3];

        if(argc != 3) {
                printf("--- Coordinate Transformation Using SOP File Pro ---\n");
                printf("Usage: %s [fpi.sop] [fpi.txt] > fpo.txt\n", argv[0]);
                exit(1);
        }

        if ((fpis = fopen (argv[1], "r")) == NULL) {
                fprintf(stderr,"Can not open %s. \n", argv[1]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpis);

        if ((fpit = fopen (argv[2], "r")) == NULL) {
                fprintf(stderr,"Can not open %s. \n", argv[2]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpit);

        i = 0;     rp = 0;
        l = 0;     p = 0;
        while (fgets(ibuf, CHR_SIZE, fpis) != NULL) {
                if(i==1) {
                        sscanf(ibuf, "%c%c%c%s", &sb[0], &sb[1], &sb[2], &in);
                        fprintf(stderr, "input 3ddfile name = ");
                        fprintf(stderr, "%s\n", in);
                }
                if(i>=5 && i<=13){
                        c[p][l] = 0.0;
                        sscanf(ibuf, "%c%c%c%c%lf", &sb[0], &sb[1], &sb[2], &sb[3], &c[p][l]);
                        //fprintf(stderr, "[%d %d] %10.8lf\n", p, l, c[p][l]);
                        l++;
                        if(l>2){
                                l = 0;
                                p ++;
                        }
                }
                if(i>=15){
                        sscanf(ibuf, "%c%c%lf", &sb[0], &sb[1], &rxyz[rp]);
                        //fprintf(stderr, "%10.3lf\n", rxyz[rp]);
                        rp ++;
                }
                i ++;
        }

        fprintf(stderr, "matrix =\n");
        for(p=0; p<3; p++){
                for(l=0; l<3; l++){
                        fprintf(stderr, "\t[%d %d] %10.8lf", p, l, c[p][l]);
                }
```

```
                fprintf(stderr, "\n");
        }
        fprintf(stderr, "Rscan position =\n");
        for(p=0; p<3; p++){
                fprintf(stderr, "\t[%d] %10.3lf\n", p, rxyz[p]);
        }

        i = 0;
        fprintf(stdout, "X\tY\tZ\tdi\tR\tG\tB\n");
        while (fgets(ibuf, CHR_SIZE, fpit) != NULL) {
                if(i>0) {
                        sscanf(ibuf, "%lf %lf %lf %d %d %d %d", &txyz[0], &txyz[1],
                            &txyz[2], &di, &r, &g, &b);
                        if(di>0){
                                for(p=0; p<3; p++){
                                        xyz[p] = 0.0;
                                        for(l=0; l<3; l++){
                                                xyz[p] = xyz[p] + c[p][l] * txyz[l];
                                        }
                                }
                                for(p=0; p<3; p++){
                                        txyz[p] = xyz[p] + rxyz[p];
                                }
                                //fprintf(stdout, "%10.3lf\t%10.3lf\t%10.3lf", txyz[0], txyz[1],
txyz[2]);
                                //fprintf(stdout, "\t%d\t%d\t%d\n", r, g, b);
                        }
                        fprintf(stdout, "%.3lf\t%.3lf\t%.3lf", txyz[0], txyz[1], txyz[2]);
                        //fprintf(stdout, "\t%d\t%d\t%d\t%d", di, r, g, b);
                        fprintf(stdout, "\n");
                }
                i++;
        }

        fclose(fpis);
        fclose(fpit);
        exit(0);
}
```

B-04. 数値標高モデル作成（point2grid.c）

```
/*-- convert for point to grid ----------------------------------------
  ﾎﾟｲﾝﾄﾃﾞｰﾀをｸﾞﾘｯﾄﾞ化する。
  DSM(表層ﾓﾃﾞﾙ)及びCRL(ｶﾗｰ画像)を作成する。
  output unit = 標高ﾃﾞｰﾀ * OFF_SIZE

  ＊y 方向の変位を見たい場合は、読み込む dz を dy に変更すると可能である。
---------------------------------- July 31. 2003 maked by misao. -*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define    MAX_SIZE  8000
#define CHR_SIZE     1024
#define DAT_SIZE     2                                  /*Data Size (byte)*/
#define OFF_SIZE     100

int        odat[MAX_SIZE][MAX_SIZE];
//int      clr[MAX_SIZE][MAX_SIZE][3];

main (argc, argv)
int argc;
char *argv[];
{
          FILE *fpi, *fpor, *fpoc;
          int        i, j, k;
          int        di, r, g, b;
          char       ibuf[CHR_SIZE];
          char       obuf[CHR_SIZE];
          double     dx, dy, dz;
          double     sx, sy;
          double     ex, ey;
          double     dint, dtmp;
          int        p_size, l_size, iz, oz;
          int        pix, lin;
          int        d_size = DAT_SIZE;

          if(argc != 8) {
                    fprintf(stderr, "--- Convert for Point to Grid Pro ---\n");
                    fprintf(stderr, "Usage: %s [i_file] [sx] [sy] [ex] [ey] [interval] [fpoDSM.raw]
//[fpoCLR.raw]\n", argv[0]);
                    exit(1);
          }

          sscanf(argv[2],"%lf", &sx);
          sscanf(argv[3],"%lf", &sy);
          sscanf(argv[4],"%lf", &ex);
          sscanf(argv[5],"%lf", &ey);
          sscanf(argv[6],"%lf", &dint);
          fprintf(stdout, "Start Coordinate(x,y): %10.5lf,\t%10.5lf\n", sx, sy);
          fprintf(stdout, "End Coordinate(x,y): %10.5lf,\t%10.5lf\n", ex, ey);
          fprintf(stdout, "Grid Interval: %10.5lf\n", dint);
          fprintf(stdout, "output unit = input * %d\n", OFF_SIZE);

          /* Calculate Image size */
          dtmp = (ex - sx) / dint;
          p_size = (int) dtmp;
          if(p_size > MAX_SIZE) {
                    fprintf(stderr, "Too Long Pixel Size -> %d\n", p_size);
                    exit(1);
          }
          dtmp = (ey - sy) / dint;
          l_size = (int) dtmp;
          if(l_size > MAX_SIZE) {
                    fprintf(stderr, "Too Long Line Size -> %d\n", l_size);
                    exit(1);
          }
          fprintf(stdout, "Image Size: %d\t%d\n", p_size, l_size);
          fprintf(stdout, "Data Size: %d\n", d_size);
```

```
/* Initialize Data */
for(i=0; i<l_size; i++) {
        for(j=0; j<p_size; j++) {
                odat[i][j] = 0;
                for(k=0; k<3; k++){
                        //clr[i][j][k] = 0;
                }
        }
}

/* Open file for reading */
if ((fpi = fopen (argv[1], "rb")) == NULL) {
        fprintf(stderr,"Can not open %s. \n", argv[1]);
        exit(1);
}
fprintf(stderr, "Open status %2d \n",fpi);

i = 0;
while (fgets(ibuf, CHR_SIZE, fpi) != NULL) {
        if(i>0) {
                sscanf(ibuf, "%lf %lf %lf %d %d %d %d", &dx, &dy, &dz, &di, &r, &g, &b);
/*dz<->dx,dy*/
                dz = dz * OFF_SIZE;
                oz = (int) dz;

                /* Calculate Pixel Line */
                dtmp = (dx - sx) / dint;
                pix = (int) dtmp;
                dtmp = (dy - sy) / dint;
                lin = l_size - (int) dtmp;

                /* Set Elevation Data */
                if(pix>=0 && pix<p_size && lin>=0 && lin<l_size) {
                        iz = odat[lin][pix];
                        if(iz==0) {
                                odat[lin][pix] = oz;
                                //clr[lin][pix][0] = r;
                                //clr[lin][pix][1] = g;
                                //clr[lin][pix][2] = b;
                                //fprintf(stderr, "%d %d %d\n", pix, lin, oz);
                        }
                        if(iz>oz) {
                                odat[lin][pix] = oz;
                                //clr[lin][pix][0] = r;
                                //clr[lin][pix][1] = g;
                                //clr[lin][pix][2] = b;
                        }
                }
        }
        i++;
}
fclose (fpi);

/* Open file for Writing */
if ((fpor = fopen (argv[7], "wb")) == NULL) {
        fprintf(stderr,"Can not open %s. \n", argv[7]);
        exit(1);
}
fprintf(stderr, "Open status %2d \n",fpor);
/*
if ((fpoc = fopen (argv[8], "wb")) == NULL) {
        fprintf(stderr,"Can not open %s. \n", argv[8]);
        exit(1);
}
fprintf(stderr, "Open status %2d \n",fpoc);
*/
for(i=0; i<l_size; i++) {
        for(j=0; j<p_size; j++) {
                xitoa(obuf, odat[i][j], d_size);
                fwrite(obuf, sizeof(char), d_size, fpor);
                /*
                for(k=0; k<3; k++){
                        putc(clr[i][j][k], fpoc);
```

```
                                }
                                */
                        }
                }
                fclose (fpor);
                //fclose (fpoc);
                exit(0);
}

xitoa(obuf, val, size)
unsigned char   *obuf;
int             val, size;
{
        int     i;
        int     itmp;

        for(i=0; i<size; i++) {
                itmp = val >> (8*(size - i - 1));
                itmp &= 0xff;
                obuf[i] = (char) itmp;
        }
        return 0;
}
```

B-05. 等高線作成（contour.c）

```
/*-- convert for point to contour ----------------------------------------------------
  ﾎﾟｲﾝﾄﾃﾞｰﾀを等高線表示するための、ﾃｷｽﾄﾃﾞｰﾀを作成する。
  ある標高のみの等高線図を作成する。
  出力されたﾃｷｽﾄﾃﾞｰﾀを ArcInfo により等高線図にする。
  ＊3dd2txt.exe を用いて出力した１方向のみのﾃﾞｰﾀを用いる。
  ＊抽出点間隔は、RscanData の line 毎。
ArcInfo
  ｺﾏﾝﾄﾞﾌﾟﾛﾝﾌﾟﾄ、ｺﾏﾝﾄﾞ arc で立ち上げる。
  Arc       : generate [作成ﾌｫﾙﾀﾞ]
  Generate : line
  ID        : &run [入力 txt]
  Generate : q
  Arc       : clean [入力ﾌｫﾙﾀﾞ] [出力ﾌｫﾙﾀﾞ] 1 1 line  (ArcView で見えるようにする)
---------------------------------------------------- August 27. 2003 made by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define    CHR_SIZE  1024
#define DAT_SIZE     80000
#define SPL_SIZE     0.2                    /*抽出幅 ele+SPL_SIZE～ele-SPL_SIZE*/
#define XXX_SIZE     1000                   /*単位を変更 [m] → [ ]*/

double     sh[DAT_SIZE], hx[DAT_SIZE], hy[DAT_SIZE], hz[DAT_SIZE];
double     sl[DAT_SIZE], lx[DAT_SIZE], ly[DAT_SIZE], lz[DAT_SIZE];
double     fx[DAT_SIZE], fy[DAT_SIZE];

main (argc, argv)
int argc;
char *argv[];
{
          FILE                         *fpi;
          unsigned char        ibuf[CHR_SIZE];
          int                                  n, i, j, k, p, e[DAT_SIZE], row;
          double                       dx, dy, dz;
          double                       ele, eleup, eledw;
          double                       hbuf=0.0, lbuf=0.0;
          double                       ix, iy, iz, idis, ie, xbuf, ybuf, fxbuf, fybuf, fxb, fyb, fdis;


          if(argc != 4){
                    printf("--- Convert for Point to Contour Pro ---\n");
                    printf("Usage: %s [fpi.txt] [ele] [row_size(1line)] > fpo.txt\n", argv[0]);
                    exit(1);
          }

          if ((fpi = fopen (argv[1], "rb")) == NULL) {
                    fprintf(stderr,"Can not open %s. \n", argv[1]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi);

          sscanf(argv[2], "%lf", &ele);
          sscanf(argv[3], "%d", &row);
          //fprintf(stderr, "start elevation:%.3lf\tend elevation:%.3lf\n", se, ee);
          fprintf(stderr, "ele:%.3lf\trow size:%d\n", ele, row);

          eleup = ele + SPL_SIZE;
          eledw = ele - SPL_SIZE;
          fprintf(stderr, "eleup:%.3lf\teledw:%.3lf\n", eleup, eledw);

          //fprintf(stdout, "x\ty\tz\n");
          n = 0;     i = 0;     j = 0;
          sh[j] = 9999.0;       sl[j] = 9999.0;
          hx[j] = 0.0;          hy[j] = 0.0;          hz[j] = 0.0;
          lx[j] = 0.0;          ly[j] = 0.0;          lz[j] = 0.0;
          while (fgets(ibuf, CHR_SIZE, fpi) != NULL) {
                    if(n>0) {
                              sscanf(ibuf, "%lf %lf %lf", &dx, &dy, &dz);
```

```
                        if(ele<=dz && eleup>dz){
                                hbuf = dz - ele;
                                if(sh[j] > hbuf){
                                        hx[j] = dx;
                                        hy[j] = dy;
                                        hz[j] = dz;
                                        sh[j] = hbuf;
                                }
                        }
                        if(eledw<=dz && ele>dz){
                                lbuf = ele - dz;
                                if(sl[j] > lbuf){
                                        lx[j] = dx;
                                        ly[j] = dy;
                                        lz[j] = dz;
                                        sl[j] = lbuf;
                                }
                        }
                        i++;
                        if(i==row){
                                /*
                                if(hz[j]!=0.0 || lz[j]!=0.0){
                                        fprintf(stdout,         "%.0lf\t%.0lf\t%.0lf\n",
hx[j]*XXX_SIZE, hy[j]*XXX_SIZE, hz[j]*XXX_SIZE);
                                        fprintf(stdout,         "%.0lf\t%.0lf\t%.0lf\n",
lx[j]*XXX_SIZE, ly[j]*XXX_SIZE, lz[j]*XXX_SIZE);
                                }
                                */
                                i = 0;
                                j ++;
                                if(j>DAT_SIZE){
                                        fprintf(stderr, "Too Long Size.\n");
                                        exit(1);
                                }
                                sh[j] = 9999.0;
                                sl[j] = 9999.0;
                                hx[j] = 0.0;    hy[j] = 0.0;    hz[j] = 0.0;
                                lx[j] = 0.0;    ly[j] = 0.0;    lz[j] = 0.0;
                        }
                }
                n++;
        }
        fprintf(stderr, "Image size:%d, %d\n", row, j);

        for(i=0; i<j; i++){
                fx[i] = 0.0;
                fy[i] = 0.0;
                e[i] = 0;
                if(hz[i]!=0.0 && lz[i]!=0.0){
                        ix = hx[i] - lx[i];
                        iy = hy[i] - ly[i];
                        iz = hz[i] - lz[i];
                        ie = hz[i] - ele;
                        xbuf = (ix * ie) / iz;
                        ybuf = (iy * ie) / iz;
                        idis = sqrt(ix*ix+iy*iy+iz*iz);
                        //fprintf(stdout, "idis:%.3lf\n", idis);
                        if(idis < SPL_SIZE*2){
                                fxbuf = hx[i] - xbuf;
                                fybuf = hy[i] - ybuf;
                                if(i>0){
                                        if(fx[i-1]!=0.0 && fy[i-1]!=0.0){
                                                fxb = fxbuf - fx[i-1];
                                                fyb = fybuf - fy[i-1];
                                                fdis = sqrt(fxb*fxb+fyb*fyb);
                                                if(fdis < SPL_SIZE*2){
                                                        fx[i] = fxbuf;
                                                        fy[i] = fybuf;
                                                        e[i] = 1;
                                                }
                                        }
                                        else{
                                                fx[i] = fxbuf;
```

```
                                        fy[i] = fybuf;
                                        e[i] = 1;
                                }
                        }
                        else{
                                fx[i] = fxbuf;
                                fy[i] = fybuf;
                                e[i] = 1;
                        }
                }
        }
        //fprintf(stdout, "%d\t%.0lf,%.0lf\n", e[i], fx[i], fy[i]);
}

/* 等高線を構成する点の出力 e[]により判別 */
ele *= XXX_SIZE;
fprintf(stdout, "%.0lf\n", ele);
p = 0;      i = 0;      k = 0;
if(e[i]==1){
        if(e[i+1]==1){
                fprintf(stdout, "%.0lf,%.0lf\n", fx[i]*XXX_SIZE, fy[i]*XXX_SIZE);
                k = 1;
                p++;
        }
}
for(i=1; i<j-1; i++){
        if(e[i]==1){
                if(e[i-1]!=0 || e[i+1]!=0){
                        fprintf(stdout, "%.0lf,%.0lf\n", fx[i]*XXX_SIZE, fy[i]*XXX_SIZE);
                        k = 1;
                        p++;
                }
        }
        if(e[i]==0){
                if(e[i-1]==1){
                        if(k==1){
                                fprintf(stdout, "end\n");
                                k = 0;
                        }
                }
        }
}
i = j-2;
if(e[i]==1){
        if(e[i-1]==1){
                fprintf(stdout, "%.0lf,%.0lf\n", fx[i]*XXX_SIZE, fy[i]*XXX_SIZE);
                k = 1;
                p++;
        }
}
if(e[i]==0){
        if(e[i-1]==1){
                if(k==1){
                        fprintf(stdout, "end\n");
                        k = 0;
                }
        }
}
fprintf(stderr, "k:%d\n", k);
if(k==1){
        fprintf(stdout, "end\n");
}
fprintf(stdout, "end");
fprintf(stderr, "p:%d\n", p);

fclose (fpi);
exit(0);
}
```

B-06. 断面図作成（traverse_bi.c）

```
/*-- convert for point to traverse (bi-linear) -------------------------------
  ﾎﾟｲﾝﾄﾃﾞｰﾀを横断線表示するための、ﾃｷｽﾄﾃﾞｰﾀを作成する。
  入力した 2 点間の断面図を作成。
  ＊3dd2txt.exe を用いて出力した 1 方向のみのﾃﾞｰﾀを用いる。
  修正点：z 値の小さい順に並び替え、小さい順に横断線を結んでいく。
        1 方向のみのﾃﾞｰﾀであれば、変な順番に結ぶことは無くなる。
ArcInfo
  ｺﾏﾝﾄﾞﾌﾟﾛﾝﾌﾟﾄ、ｺﾏﾝﾄﾞ arc で立ち上げる。
  ｺﾏﾝﾄﾞ generate [作成ﾌｫﾙﾀﾞ]
  ｺﾏﾝﾄﾞ line
  ｺﾏﾝﾄﾞ&run [入力 txt]
  ｺﾏﾝﾄﾞ q
  ｺﾏﾝﾄﾞ clean [入力ﾌｫﾙﾀﾞ] [出力ﾌｫﾙﾀﾞ] 1 1 line  (ArcView で見えるようにする)
---------------------------------------- December 20. 2003 made by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define    CHR_SIZE  1024
#define DAT_SIZE     80000
#define SPL_SIZE     0.1                    /*入力した横断位置の[+SPL_SIZE]～[-SPL_SIZE]を抽出*/
#define XXX_SIZE     1000                   /*単位を変更 [m] → [ ]*/
#define    NN        10                     /* gauss */

double     ox[DAT_SIZE], oy[DAT_SIZE], oz[DAT_SIZE];
double     ot[DAT_SIZE], od[DAT_SIZE];
int                ok[DAT_SIZE];

main (argc, argv)
int argc;
char *argv[];
{
          FILE                *fpi;
          unsigned char       ibuf[CHR_SIZE];
          double              dx, dy, dz;
          int                 di, rgb[3];
          int                 i, j, hh, ll, n, w;
          double              sx, sy, ex, ey, xdis=0.0, ydis=0.0, dist=0.0;
          double              a=0.0, b=0.0, t=0.0, l=0.0, m=0.0;
          double              fx=0.0, fy=0.0, fd=0.0, ix=0.0, iy=0.0;
          double              obuf=0.0, tt=0.0, dd=0.0, it=0.0, iz=0.0;
          int                 kbuf=0, fe;
          double              xx, yy, zz, z;
          double              aa[NN][NN], ax[NN], ab[NN];
          double              ba[NN][NN], bx[NN], bb[NN];

          if(argc != 6) {
                    printf("--- Convert for Point to Traverse (Bi-Linear) Pro ---\n");
                    printf("Usage: %s [fpi.txt] [sx] [sy] [ex] [ey] > fpo.txt\n", argv[0]);
                    exit(1);
          }

          if ((fpi = fopen (argv[1], "rb")) == NULL) {
                    fprintf(stderr,"Can not open %s. \n", argv[1]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi);

          sscanf(argv[2], "%lf", &sx);
          sscanf(argv[3], "%lf", &sy);
          sscanf(argv[4], "%lf", &ex);
          sscanf(argv[5], "%lf", &ey);
          fprintf(stderr, "sx:%.3lf\tsy:%.3lf\n", sx, sy);
          fprintf(stderr, "ex:%.3lf\tey:%.3lf\n", ex, ey);
/*
          if(sy==ey){
                    fprintf(stderr, "sy!=ey choice.");
                    exit(1);
          }*/
```

```
	xdis = ex - sx;
	ydis = ey - sy;
	dist = sqrt(xdis*xdis+ydis*ydis);
	t = 1.0;
	a = (xdis * t) / dist;
	b = (ydis * t) / dist;
	fprintf(stderr, "a:%.5lf\tb:%.5lf\n", a, b);

	l = 2.0*(a*a+b*b);

	i = 0;	n = 0;
	hh = 0;	ll = 0;
	while (fgets(ibuf, CHR_SIZE, fpi) != NULL) {
		if(i>0) {
			sscanf(ibuf, "%lf %lf %lf %d %d %d", &dx, &dy, &dz, &di, &rgb[0], &rgb[1],
&rgb[2]);
			if(di!=0){
				m = 2.0*a*(sx-dx)+2.0*b*(sy-dy);
				t = (-1.0*m) / l;
				ix = (a*t) + sx;
				iy = (b*t) + sy;
				fx = dx - ix;
				fy = dy - iy;
				fd = sqrt(fx*fx + fy*fy);
				//fprintf(stderr, "t:%.3lf\tfd:%.3lf\n", t, fd);
				if(fd<SPL_SIZE){
					ok[n] = 0;
					if(ix-dx<0){
						ok[n] = 1;		/*切断面に対して右か左
かの判別*/
						//fprintf(stdout, "%.3lf\t%.3lf\n", dx, dy);
						hh++;
					}
					else{
						//fprintf(stdout, "%.3lf\t%.3lf\n", dx, dy);
						ll++;
					}
					ox[n] = dx;
					oy[n] = dy;
					oz[n] = dz;
					ot[n] = t;
					od[n] = fd;
					n++;
					if(n>DAT_SIZE){
						fprintf(stderr, "Too Long Size.\n");
						exit(1);
					}
				}
			}
		}
		i++;
	}
	fprintf(stderr, "n:%d = ", n);
	fprintf(stderr, "hh:%d + ll:%d\n", hh, ll);

	/*並び変え*/
	for(i=1; i<n; i++){
		for(j=i; j>0; j--){
			if(oz[j]<oz[j-1]){
				obuf = ox[j];
				ox[j] = ox[j-1];
				ox[j-1] = obuf;
				obuf = oy[j];
				oy[j] = oy[j-1];
				oy[j-1] = obuf;
				obuf = oz[j];
				oz[j] = oz[j-1];
				oz[j-1] = obuf;
				kbuf = ok[j];
				ok[j] = ok[j-1];
				ok[j-1] = kbuf;
			}
```

```
                }
        }

        aa[0][0] = sx;          aa[0][1] = 1.0;
        aa[1][0] = ex;          aa[1][1] = 1.0;
        ax[0] = 0.0;            ax[1] = 0.0;
        ab[0] = sy;             ab[1] = ey;
        gauss(aa, ax, ab, 2);
        fprintf(stderr, "gauss : %.5lf, %.5lf\n", ax[0], ax[1]);

        /* 断面図を構成する点の出力 */
        fprintf(stdout, "%.0lf,%.0lf,%.0lf,%.0lf\n", sx*XXX_SIZE, sy*XXX_SIZE, ex*XXX_SIZE, ey*XXX_SIZE);
        j = 0;      w = 0;      fe = 1;
        for(i=1; i<n; i++){
                if(ok[i]!=ok[i-1]){
                        ba[0][0] = ox[i];     ba[0][1] = 1.0;
                        ba[1][0] = ox[i-1];   ba[1][1] = 1.0;
                        bx[0] = 0.0;                    bx[1] = 0.0;
                        bb[0] = oy[i];                  bb[1] = oy[i-1];
                        if(ba[0][0]!=ba[1][0]){
                                gauss(ba, bx, bb, 2);
                                xx = -(ax[1] - bx[1]) / (ax[0] - bx[0]);
                                yy = ax[0] * xx + ax[1];
                                xdis = sx - xx;
                                ydis = sy - yy;
                                dist = sqrt(xdis*xdis + ydis*ydis);       /* sx から交点までの距
離 */
                                tt = ox[i] - ox[i-1];
                                t = ox[i] - xx;
                                zz = oz[i] - oz[i-1];
                                z = oz[i] - ((zz*t) / tt);                /* 交点のz */
                                fprintf(stdout, "%.0lf,%.0lf\n", dist*XXX_SIZE, z*XXX_SIZE);
                                fe = 0;
                        }
                        else{
                                if(fe==0)
                                        fprintf(stdout, "end\n");
                                fe = 1;
                        }
                        j++;
                }
        }
        fprintf(stdout, "end\n");
        fprintf(stdout, "end");
        fprintf(stderr, "j:%d", j);

        fclose (fpi);
        exit(0);
}
gauss(a, x, b, n)
double  a[NN][NN], x[NN], b[NN];
int     n;
{
        int     i, j, k;
        double  p, g, s;

        for(k=0; k<n-1; k++){
                p = a[k][k];
                for(j=k+1; j<n; j++){
                        a[k][j] /= p;
                }
                b[k] /= p;
                for(i=k+1; i<n; i++){
                        g = a[i][k];
                        for(j=k+1; j<n; j++){
                                a[i][j] -= g * a[k][j];
                        }
                        b[i] -= g * b[k];
                }
        }
        x[n-1] = b[n-1] / a[n-1][n-1];
        for(k=n-2; k>=0; k--){
                s = b[k];
```

```
                for(j=k+1; j<n; j++){
                        s -= a[k][j] * x[j];
                }
                x[k] = s;
        }
}
```

B-07. サーフェイスモデル作成（txt2dxf.c）

```
/*-- convert for txt to dxf ----------------------------------------
  ﾚｰｻﾞｰｽｷｬﾅﾃﾞｰﾀ(***.txt)を DXF(***.dxf)に変換するﾌﾟﾛｸﾞﾗﾑ。
  (***.3dd)からではなく(***.txt)から(***.dxf)に変換するﾌﾟﾛｸﾞﾗﾑ。
  1[gon] = 0.9[deg]

  ＊ﾎﾟﾘｺﾞﾝ作成(座標入力順序)
    [1]←[0]
     ↓    ↑
    [2]→[3]
------------------------------- October 25. 2003 maked by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define    CHR_SIZE  1024
#define    FRA_SIZE  4621
#define    LIN_SIZE  1106
#define    DIS_SIZE  0.2

double     x[FRA_SIZE][LIN_SIZE];
double     y[FRA_SIZE][LIN_SIZE];
double     z[FRA_SIZE][LIN_SIZE];
int        di[FRA_SIZE][LIN_SIZE];

main (argc, argv)
int argc;
char *argv[];
{
        FILE                *fpi;
        unsigned char       ibuf[CHR_SIZE];
        int                 i, j, k, l, m, n, v;
        int                 lin, fra, l_size, polygon;
        int                 qdi, qda[4], qdb;
        double              xx[4], yy[4], zz[4];
        int                 ddi[4];
        double              bufx, bufy, bufz;
        double              dx, dy, dz, dis;
        int                 rr, gg, bb;

        if(argc != 3) {
                printf("--- Convert for TXT to DXF Pro ---\n");
                printf("Usage: %s [fpi.txt] [l_size] > fpo.dxf\n", argv[0]);
                exit(1);
        }

        if ((fpi = fopen (argv[1], "rb")) == NULL) {
                fprintf(stderr,"Can not open %s. \n", argv[1]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi);

        sscanf(argv[2], "%d", &l_size);
        fprintf(stderr, "%d\n", l_size);

        /*dxf ﾍｯﾀﾞｰ*/
        fprintf(stdout, "0\n");
        fprintf(stdout, "SECTION\n");
        fprintf(stdout, "2\n");
        fprintf(stdout, "ENTITIES\n");
        fprintf(stdout, "999\n");
        fprintf(stdout, "MISAO\n");

        n = 0;
        lin = 0;  fra = 0;
        while (fgets(ibuf, CHR_SIZE, fpi) != NULL){
```

```
                if(n>0){
                        sscanf(ibuf, "%lf %lf %lf %d %d %d %d", &x[fra][lin], &y[fra][lin],
&z[fra][lin], &di[fra][lin], &rr, &gg, &bb);
                        lin++;
                        if(lin==l_size){
                                fra ++;
                                lin = 0;
                        }
                }
                n++;
        }
        fprintf(stderr, "%d\n", fra);

        polygon = 0;
        for(i=0; i<fra-1; i++){
                /*
                if(i>100){
                        fprintf(stdout, "0\n");
                        fprintf(stdout, "ENDSEC\n");
                        fprintf(stdout, "0\n");
                        fprintf(stdout, "EOF");
                        fprintf(stderr, "%d \n", i);
                        exit(0);
                }*/
                for(j=0; j<l_size-1; j++){
                        ddi[0] = di[i][j];
                        ddi[1] = di[i+1][j];
                        ddi[2] = di[i+1][j+1];
                        ddi[3] = di[i][j+1];
                        qdi = 0;
                        for(k=0; k<4; k++){
                                if(ddi[k]>0)
                                        qdi++;
                        }
                        /* dxf 書き込み */
                        if(qdi>=3){
        /*3 点以上ﾃﾞｰﾀが存在するかﾁｪｯｸ*/
                                qdb = 0;
                                xx[0] = x[i][j];
                                yy[0] = y[i][j];
                                zz[0] = z[i][j];
                                xx[1] = x[i+1][j];
                                yy[1] = y[i+1][j];
                                zz[1] = z[i+1][j];
                                xx[2] = x[i+1][j+1];
                                yy[2] = y[i+1][j+1];
                                zz[2] = z[i+1][j+1];
                                xx[3] = x[i][j+1];
                                yy[3] = y[i][j+1];
                                zz[3] = z[i][j+1];
                                for(l=0; l<4; l++){
                                        qda[l] = 0;
                                        for(m=0; m<4; m++){
                                                if(l!=m){
                                                        dx = xx[l] - xx[m];
                                                        dy = yy[l] - yy[m];
                                                        dz = zz[l] - zz[m];
                                                        dis = sqrt(dx*dx + dy*dy + dz*dz);
                                                        /*他の点との距離が指定範囲内かﾁｪ
ｯｸ*/
                                                        if(dis<DIS_SIZE)
                                                                qda[l]++;
                                                }
                                        }
                                        if(qda[l]>=2)           /*上の条件が 2 点以上のものをﾋﾟｯｸｱ
ｯﾌﾟ*/
                                                qdb++;
                                }
                                if(qdb>=3){                     /*上の条件が 3 点以上かﾁｪｯｸ*/
                                        v = 0;
                                        /* dxf ﾃﾞｰﾀ宣言 */
                                        fprintf(stdout, "0\n");
                                        fprintf(stdout, "3DFACE\n");
```

```
                                                fprintf(stdout, "8\n");
                                                fprintf(stdout, "%d-%d\n", i, j);
                                                fprintf(stdout, "62\n");
                                                fprintf(stdout, "8\n");
                                                for(l=0; l<4; l++){
                                                        if(ddi[l]>0){
                                                                if(qda[l]>=2){
                                                                  fprintf(stdout, "%d\n", 10+v);
                                                                    fprintf(stdout,    "%.3lf\n",
xx[l]);
                                                                  fprintf(stdout, "%d\n", 20+v);
                                                                    fprintf(stdout,    "%.3lf\n",
yy[l]);
                                                                  fprintf(stdout, "%d\n", 30+v);
                                                                    fprintf(stdout,    "%.3lf\n",
zz[l]);
                                                                  bufx = xx[l];
                                                                  bufy = yy[l];
                                                                  bufz = zz[l];
                                                                  v++;
                                                                }
                                                        }
                                                }
                                                if(v == 3){
                                                        fprintf(stdout, "%d\n", 10+v);
                                                        fprintf(stdout, "%.3lf\n", bufx);
                                                        fprintf(stdout, "%d\n", 20+v);
                                                        fprintf(stdout, "%.3lf\n", bufy);
                                                        fprintf(stdout, "%d\n", 30+v);
                                                        fprintf(stdout, "%.3lf\n", bufz);
                                                }
                                                polygon++;
                                        }
                                }
                        }
                }
        }
        fprintf(stderr, "polygon : %d\n", polygon);

        /*dxf ﾌｯﾀｰ*/
        fprintf(stdout, "0\n");
        fprintf(stdout, "ENDSEC\n");
        fprintf(stdout, "0\n");
        fprintf(stdout, "EOF");

        fclose (fpi);
}
```

B-08. 画像演算（dem_opt.c）

```
/*-- dem operation ------------------------------------------------
  DEM 演算(-)
-------------------------------- September 30. 2003 made by misao. -*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define    CHR_SIZE  1024
#define    MAX_SIZE  8000
#define    BYT_SIZE  4                    /* 最大ﾃﾞｰﾀｻｲｽﾞ */
#define    OFF_SIZE  100

int                odat[MAX_SIZE][MAX_SIZE];

main (argc, argv)
int argc;
char *argv[];
{
          FILE      *fpi1, *fpi2, *fpo;
          char      obuf[BYT_SIZE];
          int       column, row;
          int       d_size, i, j, k, n;
          int       abuf[BYT_SIZE], bbuf[BYT_SIZE];
          int       cbuf, dbuf, otmp;

          if(argc != 7) {
                    fprintf(stderr, "--- Dem Operation Pro ---\n");
                    fprintf(stderr, "Usage: %s [fpi1.raw] [fpi2.raw] [column] [row] [d_size] [fpo.raw]\n",
argv[0]);
                    exit(1);
          }

          fprintf(stderr, "[%s] - [%s] = [%s]\n", argv[2], argv[1], argv[5]);
          fprintf(stderr, "[fpi1][128][fpi2]\n");

          if ((fpi1 = fopen (argv[1], "rb")) == NULL){
                    fprintf(stderr,"Can not open %s. \n", argv[1]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi1);
          if ((fpi2 = fopen (argv[2], "rb")) == NULL) {
                    fprintf(stderr,"Can not open %s. \n", argv[2]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi2);

          sscanf(argv[3],"%d", &column);
          sscanf(argv[4],"%d", &row);
          sscanf(argv[5],"%d", &d_size);

          if ((fpo = fopen (argv[6], "wb")) == NULL) {
                    fprintf(stderr,"Can not open %s. \n", argv[6]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpo);

          fprintf(stdout, "Image Size : %d\t%d\n", column, row);
          fprintf(stdout, "Data Size : %d -> 1\n", d_size);

          if(column > MAX_SIZE || row > MAX_SIZE){
                    fprintf(stderr, "Too long size, MAX_SIZE.\n");
```

```
                exit(1);
        }

        n = 0;
        for(i=0; i<column; i++){
                for(j=0; j<row; j++){
                        odat[i][j] = 0;
                        for(k=0; k<d_size; k++){
                                abuf[k] = fgetc(fpi1);
                                bbuf[k] = fgetc(fpi2);
                        }
                        cbuf = xatoi(abuf, 0, d_size);
                        dbuf = xatoi(bbuf, 0, d_size);
                        if(cbuf!=0 && dbuf!=0){
                                otmp = (dbuf - cbuf) + 128;     /*fpi2 - fpi1*/
                                if(0<=otmp && otmp<=255){
                                        odat[i][j] = otmp;
                                fprintf(stderr, "%d - %d = %d\n", dbuf, cbuf, odat[i][j]);
                                /*
                                if(0>odat[i][j] || odat[i][j]>255){
                                        fprintf(stderr, "0 > odat > 255\n");
                                        exit(1);
                                }*/
                                n++;
                                }
                        }
                }
        }
        fprintf(stderr, "operation count : %d\n", n);

        for(i=0; i<column; i++) {
                for(j=0; j<row; j++) {
                        xitoa(obuf, odat[i][j], 1);
                        fwrite(obuf, sizeof(char), 1, fpo);
                }
        }

        fclose(fpi1);
        fclose(fpi2);
        fclose(fpo);
        exit(0);
}

xatoi(ibuf, start, size)
unsigned char       *ibuf;
int                 start, size;
{
        int         i;
        int         itmp;
        int         val;

        val = 0;
        for(i=0; i<size; i++) {
                itmp = (int) ibuf[start + i];
                itmp <<= 8 * (size - i - 1);
                val += itmp;
        }
        return val;
}

xitoa(obuf, val, size)
unsigned char   *obuf;
int             val, size;
{
        int     i;
        int     itmp;

        for(i=0; i<size; i++) {
                itmp = val >> (8*(size - i - 1));
                itmp &= 0xff;
                obuf[i] = (char) itmp;
        }
        return 0;
```

```
}
```

B-09. 等高線変位量計算（contour_diff.c）

```
/*-- contour difference ------------------------------------------------
  1断面2時期分の等高線より、指定範囲(sx,sy)(ex,ey)をﾍﾙﾏｰﾄ変換し、
  指定した範囲(直線)に対して、垂直方向の変位量を抽出する。

  ＊範囲は等高線が切れてない部分を指定する。
------------------------------------ December 20. 2003 made by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define    CHR_SIZE  1024
#define XXX_SIZE     1000                /* 単位を変更 */
#define    MAX_SIZE  3000                /* 等高線構成ﾎﾟｲﾝﾄ数 */
#define    OFF_SIZE  0.05                        /* start-OFF_SIZE～end+OFF_SIZE 抽出*/
#define    NN                            10

main (argc, argv)
int argc;
char *argv[];
{
          FILE      *fpi1, *fpi2;
          char      ibuf[CHR_SIZE];
          double    dint;
          int       n, i, j, ic, jc;
          int       l, l_size;
          double    ix, iy, iz, jx, jy, jz;
          double    sx, sy, ex, ey;
          double    dtmp;
          double    dix[MAX_SIZE], diy[MAX_SIZE], djx[MAX_SIZE], djy[MAX_SIZE];
          double    xbuf, ybuf, abuf, bbuf;
          double    vi, vj;
          double    xdis, ydis, sit, dis;
          double    aa[NN][NN], xx[NN], bb[NN];

          if(argc != 8){
                    printf("--- Contour Difference Pro ---\n");
                    printf("Usage: %s [fpi1.txt] [fpi2.txt] [sx] [sy] [ex] [ey] [dint] > fpo.txt\n", argv[0]);
                    exit(1);
          }

          if ((fpi1 = fopen (argv[1], "rb")) == NULL) {
                    fprintf(stderr,"Can not open %s. \n", argv[1]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi1);

          if ((fpi2 = fopen (argv[2], "rb")) == NULL) {
                    fprintf(stderr,"Can not open %s. \n", argv[2]);
                    exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi2);

          sscanf(argv[3], "%lf", &sx);
          sscanf(argv[4], "%lf", &sy);
          sscanf(argv[5], "%lf", &ex);
          sscanf(argv[6], "%lf", &ey);
          sscanf(argv[7], "%lf", &dint);
          fprintf(stdout, "%lf, %lf, %lf, %lf, %lf\n", sx, sy, ex, ey, dint);

          /* helmert θ */
```

```
	xdis = sx - ex;
	ydis = sy - ey;
	sit = atan(ydis/xdis);

	n = 0;	i = 0;
	while (fgets(ibuf, CHR_SIZE, fpi1) != NULL){
		if(n==0){
			sscanf(ibuf, "%lf", &iz);
			iz /= XXX_SIZE;
			fprintf(stderr, "%lf\n", iz);
		}
		if(n>0){
			ix = 0.0;  iy = 0.0;
			sscanf(ibuf, "%lf,%lf", &ix, &iy);
			ix /= XXX_SIZE;
			iy /= XXX_SIZE;
			//fprintf(stderr, "%lf, %lf\n", ix, iy);
			if(ix!=0.0 && iy!=0.0){
				if(sx-OFF_SIZE<ix && ix<ex+OFF_SIZE){
					if(sy-OFF_SIZE<iy && iy<ey+OFF_SIZE){
						//fprintf(stdout, "a\t%.3lf\t%.3lf\n", ix,
iy);
						/* x,y -> transformation -> u,v */
						aa[0][0] = cos(sit); aa[0][1] = -sin(sit);
						aa[1][0] = sin(sit); aa[1][1] = cos(sit);
						xx[0] = 0.0;		xx[1] = 0.0;
						bb[0] = ix - sx;
						bb[1] = iy - sy;
						gauss(aa, xx, bb, 2);

						dix[i] = xx[0];
						diy[i] = xx[1];
						//fprintf(stdout,  "x\t%lf\t%lf\n",  xx[0],
xx[1]);
						i++;
						if(i>=MAX_SIZE){
							fprintf(stderr, "Too Long Size,
MAX_SIZE.");
							exit(1);
						}
					}
				}
			}

		}
		n++;
	}
	ic = i;

	n = 0;	j = 0;
	while (fgets(ibuf, CHR_SIZE, fpi2) != NULL) {
		if(n==0){
			sscanf(ibuf, "%lf", &jz);
			jz /= XXX_SIZE;
			fprintf(stderr, "%lf\n", jz);
		}
		if(n>0){
			jx = 0.0;  jy = 0.0;
			sscanf(ibuf, "%lf,%lf", &jx, &jy);
			jx /= XXX_SIZE;
			jy /= XXX_SIZE;
			//fprintf(stderr, "%lf, %lf\n", jx, jy);
			if(jx!=0.0 && jy!=0.0){
				if(sx-OFF_SIZE<jx && jx<ex+OFF_SIZE){
					if(sy-OFF_SIZE<jy && jy<ey+OFF_SIZE){
						//fprintf(stdout, "b\t%.3lf\t%.3lf\n", jx,
jy);
						/* x,y -> transformation -> u,v */
						aa[0][0] = cos(sit); aa[0][1] = -sin(sit);
						aa[1][0] = sin(sit); aa[1][1] = cos(sit);
						xx[0] = 0.0;		xx[1] = 0.0;
						bb[0] = jx - sx;
						bb[1] = jy - sy;
```

```
                                                        gauss(aa, xx, bb, 2);

                                                        djx[j] = xx[0];
                                                        djy[j] = xx[1];
                                                        j++;
                                                        if(j>=MAX_SIZE){
                                                                fprintf(stderr, "Too Long Size,
MAX_SIZE.");
                                                                exit(1);
                                                        }
                                                }
                                        }
                                }
                        }
                        n++;
                }
                jc = j;

                if(iz!=jz){
                        fprintf(stderr, "z : %.3lf != %.3lf\n", iz, jz);
                        exit(1);
                }

                dis = sqrt(xdis*xdis+ydis*ydis);
                dtmp = dis / dint;
                l_size = (int)dtmp;
                fprintf(stderr, "l_size : %d\n", l_size);
                fprintf(stderr, "difference = %s - %s\n\n", argv[1], argv[2]);

                fprintf(stdout, "* difference *\n");
                for(l=0; l<l_size; l++){
                        vi = 0.0;
                        for(i=0; i<ic-1; i++){
                                if(dix[i]<=(dint*l) && (dint*l)<dix[i+1]){
                                xbuf = dix[i] - dix[i+1];
                                ybuf = diy[i] - diy[i+1];
                                abuf = dix[i] - (dint*l);
                                bbuf = (abuf * ybuf) / xbuf;
                                vi = diy[i] - bbuf;
                                }
                        }
                        vj = 0.0;
                        for(j=0; j<jc-1; j++){
                                if(djx[j]<=(dint*l) && (dint*l)<djx[j+1]){
                                        xbuf = djx[j] - djx[j+1];
                                        ybuf = djy[j] - djy[j+1];
                                        abuf = djx[j] - (dint*l);
                                        bbuf = (abuf * ybuf) / xbuf;
                                        vj = djy[j] - bbuf;
                                }
                        }
                        fprintf(stdout, "%lf\t%lf\t%lf\t%lf\n", (dint*l), vi, vj, vi-vj);
                }

                fclose(fpi1);
                fclose(fpi2);
                exit(0);
}

gauss(a, x, b, n)
double  a[NN][NN];
double  x[NN];
double  b[NN];
int             n;
{
                int             i, j, k;
                double  p, g, s;
                for(k=0; k<n-1; k++){
                        p = a[k][k];
                        for(j=k+1; j<n; j++){
                                a[k][j] /= p;
                        }
                        b[k] /= p;
```

```
                for(i=k+1; i<n; i++){
                        g = a[i][k];
                        for(j=k+1; j<n; j++){
                                a[i][j] -= g * a[k][j];
                        }
                        b[i] -= g * b[k];
                }
        }
        x[n-1] = b[n-1] / a[n-1][n-1];
        for(k=n-2; k>=0; k--){
                s = b[k];
                for(j=k+1; j<n; j++){
                        s -= a[k][j] * x[j];
                }
                x[k] = s;
        }
}
```

B-10. 断面図変位量計算（traverse_diff.c）

```
/*-- traverse difference ------------------------------------------------
  1断面2時期分の断面図より、指定範囲(st,sz)(et,ez)をﾍﾙﾏｰﾄ変換し、
  指定した範囲(直線)に対して、垂直方向の変位量を抽出する。

  ＊範囲は断面図が切れてない部分を指定する。
------------------------------------ December 20. 2003 made by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define    CHR_SIZE  1024
#define XXX_SIZE     1000                /* 単位を変更 */
#define    MAX_SIZE  3000                /* 等高線構成ﾎﾟｲﾝﾄ数 */
#define    OFF_SIZE  0.05                /* start-OFF_SIZE～end+OFF_SIZE 抽出*/
#define    NN        10

main (argc, argv)
int argc;
char *argv[];
{
        FILE      *fpi1, *fpi2;
        char      ibuf[CHR_SIZE];
        double    dint;
        int               n, i, j, ic, jc;
        int               l, l_size;
        double    ix[2], iy[2], iz, it;
        double    jx[2], jy[2], jz, jt;
        double    st, sz, et, ez;
        double    dtmp;
        double    dit[MAX_SIZE], diz[MAX_SIZE], djt[MAX_SIZE], djz[MAX_SIZE];
        double    tbuf, zbuf, abuf, bbuf;
        double    vi, vj;
        double    tdis, zdis, sit, dis;
        double    aa[NN][NN], xx[NN], bb[NN];

        if(argc != 8){
                printf("--- Traverse Difference Pro ---\n");
                printf("Usage: %s [fpi1.txt] [fpi2.txt] [st] [sz] [et] [ez] [dint] > fpo.txt\n", argv[0]);
                exit(1);
        }

        if ((fpi1 = fopen (argv[1], "rb")) == NULL) {
                fprintf(stderr,"Can not open %s. \n", argv[1]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi1);

        if ((fpi2 = fopen (argv[2], "rb")) == NULL) {
                fprintf(stderr,"Can not open %s. \n", argv[2]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi2);

        sscanf(argv[3], "%lf", &st);
        sscanf(argv[4], "%lf", &sz);
```

```
        sscanf(argv[5], "%lf", &et);
        sscanf(argv[6], "%lf", &ez);
        sscanf(argv[7], "%lf", &dint);
        fprintf(stdout, "%lf, %lf, %lf, %lf, %lf\n", st, sz, et, ez, dint);

        if(st==et){
                fprintf(stderr, "st==et\n");
                exit(1);
        }
        if(sz==ez){
                fprintf(stderr, "sz==ez\n");
                exit(1);
        }

        /* helmert θ */
        tdis = st - et;
        zdis = sz - ez;
        sit = atan(zdis/tdis);

        n = 0;    i = 0;
        while (fgets(ibuf, CHR_SIZE, fpi1) != NULL){
                if(n==0){
                        sscanf(ibuf, "%lf,%lf,%lf,%lf", &ix[0], &iy[0], &ix[1], &iy[1]);
                        ix[0] /= XXX_SIZE;
                        iy[0] /= XXX_SIZE;
                        ix[1] /= XXX_SIZE;
                        iy[1] /= XXX_SIZE;
                        fprintf(stderr, "%.3lf, %.3lf -> %.3lf, %.3lf\n", ix[0], iy[0], ix[1], iy[1]);
                }
                if(n>0){
                        it = 0.0;  iz = 0.0;
                        sscanf(ibuf, "%lf,%lf", &it, &iz);
                        it /= XXX_SIZE;
                        iz /= XXX_SIZE;
                        //fprintf(stderr, "%lf, %lf\n", it, iz);
                        if(it!=0.0 && iz!=0.0){
                                //if(st-OFF_SIZE<it && it<et+OFF_SIZE){
                                        if(sz-OFF_SIZE<iz && iz<ez+OFF_SIZE){
                                                //fprintf(stderr, "a\t%.3lf\t%.3lf\n", it,
iz);
                                                /* t,z -> transformation -> u,v */
                                                aa[0][0] = cos(sit); aa[0][1] = -sin(sit);
                                                aa[1][0] = sin(sit); aa[1][1] = cos(sit);
                                                xx[0] = 0.0;         xx[1] = 0.0;
                                                bb[0] = it - st;
                                                bb[1] = iz - sz;
                                                gauss(aa, xx, bb, 2);

                                                dit[i] = xx[0];
                                                diz[i] = xx[1];
                                                //fprintf(stdout,  "x\t%lf\t%lf\n",  xx[0],
xx[1]);
                                                i++;
                                                if(i>=MAX_SIZE){
                                                        fprintf(stderr, "Too Long Size,
MAX_SIZE.");
                                                        exit(1);
                                                }
                                        }
                                //}
                        }

                }
                n++;
        }
        ic = i;

        n = 0;    j = 0;
        while (fgets(ibuf, CHR_SIZE, fpi2) != NULL) {
                if(n==0){
                        sscanf(ibuf, "%lf,%lf,%lf,%lf", &jx[0], &jy[0], &jx[1], &jy[1]);
                        jx[0] /= XXX_SIZE;
                        jy[0] /= XXX_SIZE;
```

```
                        jx[1] /= XXX_SIZE;
                        jy[1] /= XXX_SIZE;
                        fprintf(stderr, "%.3lf, %.3lf -> %.3lf, %.3lf\n", jx[0], jy[0], jx[1], jy[1]);
                }
                if(n>0){
                        jt = 0.0;  jz = 0.0;
                        sscanf(ibuf, "%lf,%lf", &jt, &jz);
                        jt /= XXX_SIZE;
                        jz /= XXX_SIZE;
                        //fprintf(stderr, "%lf, %lf\n", jt, jz);
                        if(jt!=0.0 && jz!=0.0){
                                //if(st-OFF_SIZE<jt && jt<et+OFF_SIZE){
                                        if(sz-OFF_SIZE<jz && jz<ez+OFF_SIZE){
                                                //fprintf(stdout, "b\t%.3lf\t%.3lf\n", jt,
jz);
                                                /* t,z -> transformation -> u,v */
                                                aa[0][0] = cos(sit); aa[0][1] = -sin(sit);
                                                aa[1][0] = sin(sit); aa[1][1] = cos(sit);
                                                xx[0] = 0.0;         xx[1] = 0.0;
                                                bb[0] = jt - st;
                                                bb[1] = jz - sz;
                                                gauss(aa, xx, bb, 2);

                                                djt[j] = xx[0];
                                                djz[j] = xx[1];
                                                j++;
                                                if(j>=MAX_SIZE){
                                                        fprintf(stderr, "Too Long Size,
MAX_SIZE.");
                                                        exit(1);
                                                }
                                        }
                                //}
                        }
                }
                n++;
        }
        jc = j;

        if(ix[0]!=jx[0] || iy[0]!=jy[0] || ix[1]!=jx[1] || iy[1]!=jy[1]){
                fprintf(stderr, "\nx1,y1 : %.3lf, %.3lf != %.3lf, %.3lf\n", ix[0], iy[0], jx[0], jy[0]);
                fprintf(stderr, "x2,y2 : %.3lf, %.3lf != %.3lf, %.3lf\n", ix[1], iy[1], jx[1], jy[1]);
                exit(1);
        }

        dis = sqrt(tdis*tdis+zdis*zdis);
        dtmp = dis / dint;
        l_size = (int)dtmp;
        fprintf(stderr, "l_size : %d\n", l_size);
        fprintf(stderr, "difference = %s - %s\n\n", argv[1], argv[2]);

        fprintf(stdout, "* difference *\n");
        for(l=0; l<l_size; l++){
                vi = 0.0;
                for(i=0; i<ic-1; i++){
                        if(dit[i]<=(dint*l) && (dint*l)<dit[i+1]){
                        tbuf = dit[i] - dit[i+1];
                        zbuf = diz[i] - diz[i+1];
                        abuf = dit[i] - (dint*l);
                        bbuf = (abuf * zbuf) / tbuf;
                        vi = diz[i] - bbuf;
                        }
                }
                vj = 0.0;
                for(j=0; j<jc-1; j++){
                        if(djt[j]<=(dint*l) && (dint*l)<djt[j+1]){
                                tbuf = djt[j] - djt[j+1];
                                zbuf = djz[j] - djz[j+1];
                                abuf = djt[j] - (dint*l);
                                bbuf = (abuf * zbuf) / tbuf;
                                vj = djz[j] - bbuf;
                        }
                }
```

```
                fprintf(stdout, "%lf\t%lf\t%lf\t%lf\n", (dint*l), vi, vj, vi-vj);
        }

        fclose(fpi1);
        fclose(fpi2);
        exit(0);
}

gauss(a, x, b, n)
double  a[NN][NN], x[NN], b[NN];
int     n;
{
        int     i, j, k;
        double  p, g, s;

        for(k=0; k<n-1; k++){
                p = a[k][k];
                for(j=k+1; j<n; j++){
                        a[k][j] /= p;
                }
                b[k] /= p;
                for(i=k+1; i<n; i++){
                        g = a[i][k];
                        for(j=k+1; j<n; j++){
                                a[i][j] -= g * a[k][j];
                        }
                        b[i] -= g * b[k];
                }
        }
        x[n-1] = b[n-1] / a[n-1][n-1];

        for(k=n-2; k>=0; k--){
                s = b[k];
                for(j=k+1; j<n; j++){
                        s -= a[k][j] * x[j];
                }
                x[k] = s;
        }
}
```

B-11. グリッド上のサーフェイス演算（surface2grid.c）

```
/*-- convert for surface difference to grid -----------------------------
  ｸﾞﾘｯﾄﾞ上での演算ﾌﾟﾛｸﾞﾗﾑ(変位量(cm) = 2 時期目 - 1 時期目)
  dat[men_size][point_size][xyz]

  ＊ｸﾞﾘｯﾄﾞ交点から伸ばしたﾍﾞｸﾄﾙが、面を構成する辺と交わった場合、
    どちらの面に属する点(交点)かの判別も行わなくてはならない。
    しかし、その判別ｱﾙｺﾞﾘｽﾞﾑはとても複雑になることが予想されることと、
    そのような点が存在する可能性は非常に低いため、用いないこととする。

  ＊ｻｰﾌｪｲｽ構成点の読込み順位
  dat[1]←dat[0]
    ↓       ↑
  dat[2]→dat[3]
  4 点で構成されている場合、[0][1][2]&[2][3][0]の 3 点ずつ行なう。
------------------------------------- October. 2003 maked by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define    CHR_SIZE 1024
#define    MAX_SIZE 200000
#define    COL_SIZE 5000
#define    ROW_SIZE 3000
#define    BYT_SIZE 1                    /* data size (byte) */
#define    NN 3                          /* gauss */
#define    CMM_SIZE 100                  /* 100:cm, 1000:mm の変位を読取る */
#define    SC_SIZE 1000                  /* 最大抽出面数 */
#define    DD_SIZE 1                     /* ｸﾞﾘｯﾄﾞ交点から面の重心までの距離 */

double     dat1[MAX_SIZE][4][3];
double     dat2[MAX_SIZE][4][3];
int        grid[ROW_SIZE][COL_SIZE];

xitoa(obuf, val, size)
unsigned char   *obuf;
int             val, size;
{
        int     i;
        int     itmp;

        for(i=0; i<size; i++) {
                itmp = val >> (8*(size - i - 1));
                itmp &= 0xff;
                obuf[i] = (char) itmp;
        }
        return 0;
}

gauss(a, x, b, n)
double     a[NN][NN], x[NN], b[NN];
```

```
int         n;
{
            int         i, j, k;
            double      p, g, s;

            for(k=0; k<n-1; k++){
                        p = a[k][k];
                        for(j=k+1; j<n; j++){
                                    a[k][j] /= p;
                        }
                        b[k] /= p;
                        for(i=k+1; i<n; i++){
                                    g = a[i][k];
                                    for(j=k+1; j<n; j++){
                                                a[i][j] -= g * a[k][j];
                                    }
                                    b[i] -= g * b[k];
                        }
            }
            x[n-1] = b[n-1] / a[n-1][n-1];

            for(k=n-2; k>=0; k--){
                        s = b[k];
                        for(j=k+1; j<n; j++){
                                    s -= a[k][j] * x[j];
                        }
                        x[k] = s;
            }
}

/* 面の式の係数を導く */
lsq(dx, dy, dz, xx, n)
double      dx[3], dy[3], dz[3];
double      xx[NN];
int                     n;
{
            double      aa[NN][NN], bb[NN];

            aa[0][0] = dx[0];     aa[0][1] = dy[0];     aa[0][2] = dz[0];
            aa[1][0] = dx[1];     aa[1][1] = dy[1];     aa[1][2] = dz[1];
            aa[2][0] = dx[2];     aa[2][1] = dy[2];     aa[2][2] = dz[2];
            bb[0] = 1.0;
            bb[1] = 1.0;
            bb[2] = 1.0;

            gauss(aa, xx, bb, 3);
}

/* 直線の式よりｸﾞﾘｯﾄﾞ交点が面内にあるか判別する */
int suf_search(dx, dz, ix, iz)
double      dx[3], dz[3];
double      ix, iz;
{
            int                     i, n, suf, f;
            double      a[NN][NN], x[NN], b[NN];
            double      buf, buff, tmp;
            double      fx[5], fz[5];

            f = 0;
            fx[0] = dx[0];        fz[0] = dz[0];
            fx[1] = dx[1];        fz[1] = dz[1];
            fx[2] = dx[2];        fz[2] = dz[2];
            fx[3] = dx[0];        fz[3] = dz[0];
            fx[4] = dx[1];        fz[4] = dz[1];

            suf = 0;   n = 3;
            for(i=0; i<n; i++){
                        a[0][0] = fx[i];      a[0][1] = 1.0;
                        a[1][0] = fx[i+1];    a[1][1] = 1.0;
                        b[0] = fz[i];
                        b[1] = fz[i+1];
                        x[0] = 0.0;
                        x[1] = 0.0;
```

```
                tmp = a[0][0]*a[1][1] - a[0][1]*a[1][0];
                if(tmp == 0.0)                  /* 対角行列が等しい時 */
                        break;

                gauss(a, x, b, 2);

                buf = fx[i+2]*x[0] + x[1] - fz[i+2];
                buff = ix*x[0] + x[1] - iz;

                if(buf > 0 && buff > 0)
                        suf++;
                if(buf < 0 && buff < 0)
                        suf++;
        }

        if(suf == n)
                f = n;
        else
                f = 0;

        return(f);
}

main(argc, argv)
int argc;
char *argv[];
{
        FILE    *fpi1, *fpi2, *fpo;
        char    ibuf[CHR_SIZE], obuf[CHR_SIZE];
        double  sx, sz, ex, ez, dint;
        double  dtmp, dmin;
        int             l_size, m_size, gbuf;
        int             i, j, l, m, k, n, p, o, f;
        int             mc1, mc2, cgrid, byte_s;
        int             h_size, p_size, suf;
        double  ix, iz;
        double  ay[SC_SIZE], by[SC_SIZE];
        double  min[3], max[3];
        int             ac, bc, sn;
        double  px[3], py[3], pz[3], xx[NN];
        double  ox[5], oy[5], oz[5];
        double  avex, avez, dd;

        if(argc != 9){
                printf("--- Convert for Surface Difference to Grid Pro ---\n");
                printf("Usage:%s   [fpi1.dxf]   [fpi2.dxf]   [sx]   [sz]   [ex]   [ez]   [interval]
[fpo.raw]\n",argv[0]);
                exit(1);
        }

        sscanf(argv[3], "%lf", &sx);
        sscanf(argv[4], "%lf", &sz);
        sscanf(argv[5], "%lf", &ex);
        sscanf(argv[6], "%lf", &ez);
        sscanf(argv[7], "%lf", &dint);

        /* grid size */
        dtmp = (ex - sx) / dint;
        l_size = (int) dtmp;
        dtmp = (ez - sz) / dint;
        m_size = (int) dtmp;
        fprintf(stderr, "\n--Output Image--\n");
        fprintf(stderr, "data size : %d byte\n", BYT_SIZE);
        fprintf(stderr, "column, row : %d, %d\n", l_size, m_size);
        fprintf(stderr, "left up coner coordinates (sx, ez) : %.3lf, %.3lf\n", sx, ez);
        fprintf(stderr, "unit : inuput data * %d\n\n", CMM_SIZE);

        h_size = 6;                             /* DXF heder size */
        p_size = 12;                    /* surface point * xyz(3) */

        if((fpi1 = fopen (argv[1],"rb")) == NULL){
                fprintf(stderr,"Can not open %s.\n",argv[1]);
```

```
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi1);

        /*ﾍｯﾀﾞｰ読み込み*/
        for(i=0; i<h_size*2; i++)
                fgets(ibuf, CHR_SIZE, fpi1);

        n=0;        p=0;
        o=0;        mc1=0;      suf=0;
        for(i=0; i<3; i++){
                min[i] = 10000000.0;
                max[i] = -10000000.0;
        }
        while (fgets(ibuf, CHR_SIZE, fpi1) != NULL){
                if(n>0){
                        sscanf(ibuf, "%lf", &dat1[mc1][p][o]);
                        max[o] = dat1[mc1][p][o] > max[o] ? dat1[mc1][p][o] : max[o];
                        min[o] = dat1[mc1][p][o] < min[o] ? dat1[mc1][p][o] : min[o];
                        n = -1;
                        o++;        suf++;
                        if(o>2){
                                o=0;        p++;
                                if(p>3){
                                        p=0;        mc1++;
                                        if(mc1>=MAX_SIZE){
                                                fprintf(stderr,     "too     long     size,
MAX_SIZE.\n");
                                                exit(1);
                                        }
                                }
                        }
                        if(suf>p_size-1){
                                for(i=0; i<h_size; i++){
                                        fgets(ibuf, CHR_SIZE, fpi1);
                                }
                                suf=0;
                        }
                }
                n++;
        }
        fprintf(stderr, "fpi1\tx\ty\tz\n");
        fprintf(stderr, "min\t%.3lf\t%.3lf\t%.3lf\n", min[0], min[1], min[2]);
        fprintf(stderr, "max\t%.3lf\t%.3lf\t%.3lf\n", max[0], max[1], max[2]);
        fprintf(stderr, "surface count : %d\n\n", mc1);

        if((fpi2 = fopen (argv[2],"rb")) == NULL){
                fprintf(stderr,"Can not open %s.\n",argv[2]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi2);

        /*ﾍｯﾀﾞｰ読み込み*/
        for(i=0; i<h_size*2; i++)
                fgets(ibuf, CHR_SIZE, fpi2);

        n=0;        p=0;
        o=0;        mc2=0;      suf=0;
        for(i=0; i<3; i++){
                min[i] = 10000000.0;
                max[i] = -10000000.0;
        }
        while (fgets(ibuf, CHR_SIZE, fpi2) != NULL){
                if(n>0){
                        sscanf(ibuf, "%lf", &dat2[mc2][p][o]);
                        max[o] = dat2[mc2][p][o] > max[o] ? dat2[mc2][p][o] : max[o];
                        min[o] = dat2[mc2][p][o] < min[o] ? dat2[mc2][p][o] : min[o];
                        n = -1;
                        o++;        suf++;
                        if(o>2){
                                o=0;        p++;
                                if(p>3){
                                        p=0;        mc2++;
```

```
                                        if(mc2>=MAX_SIZE){
                                                fprintf(stderr,     "too     long     size,
MAX_SIZE.\n");
                                                exit(1);
                                        }
                                }
                        }
                        if(suf>p_size-1){
                                for(i=0; i<h_size; i++){
                                        fgets(ibuf, CHR_SIZE, fpi2);
                                }
                                suf=0;
                        }
                }
                n++;
        }
        fprintf(stderr, "fpi2\tx\ty\tz\n");
        fprintf(stderr, "min\t%.3lf\t%.3lf\t%.3lf\n", min[0], min[1], min[2]);
        fprintf(stderr, "max\t%.3lf\t%.3lf\t%.3lf\n", max[0], max[1], max[2]);
        fprintf(stderr, "surface count : %d\n\n", mc2);

        byte_s = 256;
        for(k=0; k<BYT_SIZE-1; k++){
                byte_s = byte_s * 256;
        }
        /* 探索 */
        fprintf(stderr, "change = %s - %s\n", argv[1], argv[2]);
        cgrid = 0;
        for(i=0; i<m_size; i++){
                iz = ez - (dint*i) + (dint/2);                          /* ｸﾞﾘｯﾄﾞ交点 z座標 */
                for(j=0; j<l_size; j++){
                        ac = 0;
                        bc = 0;
                        grid[i][j] = 0;                                 /* 初期値 */
                        ix = sx + (dint*j) + (dint/2);                  /* ｸﾞﾘｯﾄﾞ交点 x座標 */
                        for(l=0; l<mc1; l++){
                                sn = 0;
                                if(dat1[l][2][0]==dat1[l][3][0] && dat1[l][2][1]==dat1[l][3][1]
&& dat1[l][2][2]==dat1[l][3][2]){       /* 3 点目==4 点目? */
                                  sn = 1;
                                   ox[0] = dat1[l][0][0];   oy[0] = dat1[l][0][1];   oz[0] =
dat1[l][0][2];
                                   ox[1] = dat1[l][1][0];   oy[1] = dat1[l][1][1];   oz[1] =
dat1[l][1][2];
                                   ox[2] = dat1[l][2][0];   oy[2] = dat1[l][2][1];   oz[2] =
dat1[l][2][2];
                                }
                                else{
                                  sn = 2;
                                   ox[0] = dat1[l][0][0];   oy[0] = dat1[l][0][1];   oz[0] =
dat1[l][0][2];
                                   ox[1] = dat1[l][1][0];   oy[1] = dat1[l][1][1];   oz[1] =
dat1[l][1][2];
                                   ox[2] = dat1[l][2][0];   oy[2] = dat1[l][2][1];   oz[2] =
dat1[l][2][2];
                                   ox[3] = dat1[l][3][0];   oy[3] = dat1[l][3][1];   oz[3] =
dat1[l][3][2];
                                   ox[4] = dat1[l][0][0];   oy[4] = dat1[l][0][1];   oz[4] =
dat1[l][0][2];
                                }
                                for(m=0; m<=sn; m=m+2){
                                  ay[ac] = 0.0;
                                  px[0] = ox[0+m];   px[1] = ox[1+m];     px[2] = ox[2+m];
                                  py[0] = oy[0+m];   py[1] = oy[1+m];     py[2] = oy[2+m];
                                  pz[0] = oz[0+m];   pz[1] = oz[1+m];     pz[2] = oz[2+m];
                                  avex = (px[0] + px[1] + px[2]) / 3;
                                  avez = (pz[0] + pz[1] + pz[2]) / 3;
                                  dd = (ix - avex)*(ix - avex) + (iz - avez)*(iz - avez);
                                  if(dd < DD_SIZE){
                                    f = suf_search(px, pz, ix, iz);         /* f = 0 or 3 */
                                    if(f>0){
                                      xx[0] = 0.0;  xx[1] = 0.0;  xx[2] = 0.0;
                                      lsq(px, py, pz, xx, f);
```

```
                                        ay[ac] = (-xx[0]*ix -xx[2]*iz +1.0) / xx[1];

                                        //for(k=0; k<3; k++)
                                          //fprintf(stdout, "%.3lf, %.3lf, %.3lf\n", px[k], py[k],
pz[k]);
                                         //fprintf(stdout, "%.3lf,%.3lf | f=%d, xx=%lf, %lf, %lf
ay[%d]=%.3lf\n", ix, iz, f, xx[0], xx[1], xx[2], ac, ay[ac]);
                                        ac++;
                                        if(ac>=SC_SIZE){
                                          fprintf(stderr, "too long size, SC_SIZE.\n");
                                          exit(1);
                                        }
                                      }
                                    }
                                  }
                                }
                              }
                              //fprintf(stdout, "acount : %d\n", ac);
                              for(l=0; l<mc2; l++){
                                sn = 0;
                                if(dat2[l][2][0]==dat2[l][3][0] && dat2[l][2][1]==dat2[l][3][1]
&& dat2[l][2][2]==dat2[l][3][2]){       /* 3 点目==4 点目? */
                                  sn = 1;
                                   ox[0] = dat2[l][0][0];   oy[0] = dat2[l][0][1];   oz[0] =
dat2[l][0][2];
                                   ox[1] = dat2[l][1][0];   oy[1] = dat2[l][1][1];   oz[1] =
dat2[l][1][2];
                                   ox[2] = dat2[l][2][0];   oy[2] = dat2[l][2][1];   oz[2] =
dat2[l][2][2];
                                }
                                else{
                                  sn = 2;
                                   ox[0] = dat2[l][0][0];   oy[0] = dat2[l][0][1];   oz[0] =
dat2[l][0][2];
                                   ox[1] = dat2[l][1][0];   oy[1] = dat2[l][1][1];   oz[1] =
dat2[l][1][2];
                                   ox[2] = dat2[l][2][0];   oy[2] = dat2[l][2][1];   oz[2] =
dat2[l][2][2];
                                   ox[3] = dat2[l][3][0];   oy[3] = dat2[l][3][1];   oz[3] =
dat2[l][3][2];
                                   ox[4] = dat2[l][0][0];   oy[4] = dat2[l][0][1];   oz[4] =
dat2[l][0][2];
                                }
                                for(m=0; m<=sn; m=m+2){
                                          by[bc] = 0.0;
                                          px[0] = ox[0+m];     px[1] = ox[1+m];     px[2]      =
ox[2+m];
                                          py[0] = oy[0+m];     py[1] = oy[1+m];     py[2]      =
oy[2+m];
                                          pz[0] = oz[0+m];     pz[1] = oz[1+m];     pz[2]      =
oz[2+m];
                                          avex = (px[0] + px[1] + px[2]) / 3;
                                          avez = (pz[0] + pz[1] + pz[2]) / 3;
                                          dd = (ix - avex)*(ix - avex) + (iz - avez)*(iz - avez);
                                          if(dd < DD_SIZE){
                                            f = suf_search(px, pz, ix, iz);          /* f = 0 or 3
*/
                                            if(f>0){
                                              xx[0] = 0.0;  xx[1] = 0.0;  xx[2] = 0.0;
                                              lsq(px, py, pz, xx, f);
                                              by[bc] = (-xx[0]*ix -xx[2]*iz +1.0) / xx[1];

                                               //fprintf(stdout,     "%.3lf,%.3lf     |     f=%d,
xx=%lf, %lf, %lf by[%d]=%.3lf\n", ix, iz, f, xx[0], xx[1], xx[2], bc, by[bc]);
                                              bc++;
                                              if(bc>=SC_SIZE){
                                                fprintf(stderr, "too long size, SC_SIZE.\n");
                                                exit(1);
                                              }
                                            }
                                          }
                                }
                              }
                              if(ac>0 && bc>0){
```

```
                                        dmin = 99999.0;
                                        for(l=0; l<ac; l++){
                                                for(m=0; m<bc; m++){
                                                        dtmp = (ay[l] - by[m]) * CMM_SIZE;
                                                        dmin = dtmp < dmin ? dtmp : dmin;
                                                }
                                        }
                                        //fprintf(stdout, "dmin = %lf\n", dmin);
                                        gbuf = (int)dmin + (int)(byte_s / 2);
                                        if(0 <= gbuf && gbuf <=byte_s-1){
                                                grid[i][j] = gbuf;
                                                cgrid++;
                                        }
                                        /*
                                        if(0 > grid[i][j] || grid[i][j] > byte_s-1){
                                                fprintf(stderr,   "grid[%d][%d]   =   %d\n",   i,   j,
grid[i][j]);
                                                fprintf(stderr, "too small size, CMM_SIZE.\n");
                                                exit(1);
                                        }*/
                                }
                        }
                }
                fprintf(stderr, "change point : %d\n", cgrid);

                if((fpo = fopen (argv[8],"wb")) == NULL){
                        fprintf(stderr,"Can not open %s.\n",argv[8]);
                        exit(1);
                }
                fprintf(stderr, "Open status %2d \n", fpo);

                for(i=0; i<m_size; i++){
                        for(j=0; j<l_size; j++){
                                xitoa(obuf, grid[i][j], BYT_SIZE);
                                fwrite(obuf, sizeof(char), BYT_SIZE, fpo);
                        }
                }

                fclose(fpi1);
                fclose(fpi2);
                fclose(fpo);
                exit(0);
}
```

B-12. 最近隣点探索による変位量ベクトル化（surface2vector.c）

```
/*-- convert for surface difference to vector --------------------------------
  作成したDXFより各面の重心を求め、最も近い重心同士を用いﾍﾞｸﾄﾙ化する。
  ＊変位量(cm) = 2時期目 - 1時期目
----------------------------------------- October 30. 2003 maked by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define    CHR_SIZE 1024
#define    MAX_SIZE 800000
#define    F_SIZE 0.5          /* 変位量閾値 */

double     dat1[MAX_SIZE][4][3];
double     dat2[MAX_SIZE][4][3];

main(argc, argv)
int argc;
char *argv[];
{
          FILE      *fpi1, *fpi2;
          char      ibuf[CHR_SIZE];
          int       h_size, p_size;
          int       p, o, n, mc1, mc2, suf;
          int       i, j, k, fi, fj;
          double    min[3], max[3];
          double    xtmp[4], ytmp[4], ztmp[4];
```

```
        double    xbuf[4], ybuf[4], zbuf[4];
        double    xsum[2], ysum[2], zsum[2];
        double    xave[2], yave[2], zave[2];
        double    xd, yd, zd, dis, sdis;
        double    xx, yy, zz;

        if(argc != 3){
                printf("--- Convert for Surface Difference to Vector Pro ---\n");
                printf("Usage:%s [fpi1.dxf] [fpi2.dxf] > vector.dxf\n",argv[0]);
                exit(1);
        }

        h_size = 6;
        p_size = 12;

        /* input DXF file 1 */
        if((fpi1 = fopen (argv[1],"rb")) == NULL){
                fprintf(stderr,"Can not open %s.\n",argv[1]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi1);

        /*ﾍｯﾀﾞｰ読み込み*/
        for(i=0; i<h_size*2; i++)
                fgets(ibuf, CHR_SIZE, fpi1);

        n=0;      p=0;
        o=0;      mc1=0;    suf=0;
        for(i=0; i<3; i++){
                min[i] = 10000000.0;
                max[i] = -10000000.0;
        }
        while (fgets(ibuf, CHR_SIZE, fpi1) != NULL){
                if(n>0){
                        sscanf(ibuf, "%lf", &dat1[mc1][p][o]);
                        max[o] = dat1[mc1][p][o] > max[o] ? dat1[mc1][p][o] : max[o];
                        min[o] = dat1[mc1][p][o] < min[o] ? dat1[mc1][p][o] : min[o];
                        n = -1;
                        o++;      suf++;
                        if(o>2){
                                o=0;      p++;
                                if(p>3){
                                        p=0;      mc1++;
                                        if(mc1>=MAX_SIZE){
                                                fprintf(stderr,     "too     long     size,
MAX_SIZE.\n");
                                                exit(1);
                                        }
                                }
                        }
                        if(suf>p_size-1){
                                for(i=0; i<h_size; i++){
                                        fgets(ibuf, CHR_SIZE, fpi1);
                                }
                                suf=0;
                        }
                }
                n++;
        }
        fprintf(stderr, "fpi1 surface : %d\n", mc1);
        fprintf(stderr, "\tx\ty\tz\n");
        fprintf(stderr, "min\t%.3lf\t%.3lf\t%.3lf\n", min[0], min[1], min[2]);
        fprintf(stderr, "max\t%.3lf\t%.3lf\t%.3lf\n", max[0], max[1], max[2]);

        /* input DXF file 2 */
        if((fpi2 = fopen (argv[2],"rb")) == NULL){
                fprintf(stderr,"Can not open %s.\n",argv[2]);
                exit(1);
        }
        fprintf(stderr, "Open status %2d \n",fpi2);

                /*ﾍｯﾀﾞｰ読み込み*/
        for(i=0; i<h_size*2; i++)
```

```
		fgets(ibuf, CHR_SIZE, fpi2);

	n=0;	p=0;
	o=0;	mc2=0;	suf=0;
	for(i=0; i<3; i++){
		min[i] = 10000000.0;
		max[i] = -10000000.0;
	}
	while (fgets(ibuf, CHR_SIZE, fpi2) != NULL){
		if(n>0){
			sscanf(ibuf, "%lf", &dat2[mc2][p][o]);
			max[o] = dat2[mc2][p][o] > max[o] ? dat2[mc2][p][o] : max[o];
			min[o] = dat2[mc2][p][o] < min[o] ? dat2[mc2][p][o] : min[o];
			n = -1;
			o++;	suf++;
			if(o>2){
				o=0;	p++;
				if(p>3){
					p=0;	mc2++;
					if(mc2>=MAX_SIZE){
						fprintf(stderr,	"too	long	size,
MAX_SIZE.\n");
						exit(1);
					}
				}
			}
			if(suf>p_size-1){
				for(i=0; i<h_size; i++){
					fgets(ibuf, CHR_SIZE, fpi2);
				}
				suf=0;
			}
		}
		n++;
	}
	fprintf(stderr, "fpi2 surface : %d\n", mc2);
	fprintf(stderr, "\tx\ty\tz\n");
	fprintf(stderr, "min\t%.3lf\t%.3lf\t%.3lf\n", min[0], min[1], min[2]);
	fprintf(stderr, "max\t%.3lf\t%.3lf\t%.3lf\n", max[0], max[1], max[2]);

	fprintf(stdout, "0\n");
	fprintf(stdout, "SECTION\n");
	fprintf(stdout, "2\n");
	fprintf(stdout, "ENTITIES\n");
	fprintf(stdout, "999\n");
	fprintf(stdout, "maked by misao.\n");
	for(i=0; i<mc1; i++){
		sdis = 10000000.0;
		for(k=0; k<4; k++){
			xtmp[k] = dat1[i][k][0];
			ytmp[k] = dat1[i][k][1];
			ztmp[k] = dat1[i][k][2];
		}
		if(xtmp[2]!=xtmp[3] || ytmp[2]!=ytmp[3] || ztmp[2]!=ztmp[3])
			fi = 4;
		else
			fi = 3;
		xsum[0] = 0.0;
		ysum[0] = 0.0;
		zsum[0] = 0.0;
		for(k=0; k<fi; k++){
			xsum[0] += xtmp[k];
			ysum[0] += ytmp[k];
			zsum[0] += ztmp[k];
		}
		xave[0] = xsum[0]/fi;
		yave[0] = ysum[0]/fi;
		zave[0] = zsum[0]/fi;
		for(j=0; j<mc2; j++){
			for(k=0; k<4; k++){
				xbuf[k] = dat2[j][k][0];
				ybuf[k] = dat2[j][k][1];
				zbuf[k] = dat2[j][k][2];
```

```
                        }
                        if(xbuf[2]!=xbuf[3] || ybuf[2]!=ybuf[3] || zbuf[2]!=zbuf[3])
                                fj = 4;
                        else
                                fj = 3;
                        xsum[1] = 0.0;
                        ysum[1] = 0.0;
                        zsum[1] = 0.0;
                        for(k=0; k<fj; k++){
                                xsum[1] += xbuf[k];
                                ysum[1] += ybuf[k];
                                zsum[1] += zbuf[k];
                        }
                        xave[1] = xsum[1]/fj;
                        yave[1] = ysum[1]/fj;
                        zave[1] = zsum[1]/fj;

                        xd = xave[1] - xave[0];
                        yd = yave[1] - yave[0];
                        zd = zave[1] - zave[0];
                        dis = sqrt(xd*xd + yd*yd + zd*zd);

                        if(sdis>dis){
                                sdis = dis;
                                xx = xave[1];
                                yy = yave[1];
                                zz = zave[1];
                        }
                }
                if(sdis<F_SIZE){
                        if(xave[0]!=xx || yave[0]!=yy || zave[0]!=zz){
                                fprintf(stdout, "0\n");
                                fprintf(stdout, "LINE\n");
                                fprintf(stdout, "5\n");
                                fprintf(stdout, "28\n");
                                fprintf(stdout, "100\n");
                                fprintf(stdout, "AcDbEntity\n");
                                fprintf(stdout, "8\n");
                                fprintf(stdout, "0\n");
                                fprintf(stdout, "100\n");
                                fprintf(stdout, "AcDbLine\n");
                                fprintf(stdout, "10\n");
                                fprintf(stdout, "%.3lf\n", xave[0]);
                                fprintf(stdout, "20\n");
                                fprintf(stdout, "%.3lf\n", yave[0]);
                                fprintf(stdout, "30\n");
                                fprintf(stdout, "%.3lf\n", zave[0]);
                                fprintf(stdout, "11\n");
                                fprintf(stdout, "%.3lf\n", xx);
                                fprintf(stdout, "21\n");
                                fprintf(stdout, "%.3lf\n", yy);
                                fprintf(stdout, "31\n");
                                fprintf(stdout, "%.3lf\n", zz);
                                //fprintf(stderr, "%.3lf\n", sdis);
                        }
                }
        }
        fprintf(stdout, "0\n");
        fprintf(stdout, "ENDSEC\n");
        fprintf(stdout, "0\n");
        fprintf(stdout, "EOF\n");

        fclose(fpi1);
        fclose(fpi2);
        exit(0);
}
```

B-13. 対象物の自動抽出（point2surface.c）

```
/*-- convert for point to surface --------------------------------
  ある面の中心座標(概ね)を指定し、指定面のﾎﾟｲﾝﾄﾃﾞｰﾀを抽出する。

  *面の式  ax + by + cz + 1 = 0
  *ある点までの距離(d = -1)
  dis = |ax + by + cz + d| / (a^2 + b^2 + c^2)

  *面を 4 つに分けて導く
---------------------------- October 24. 2003 maked by misao. -*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define    FRA_SIZE  4621       /* Rscan, max frame size*/
#define    LIN_SIZE  1106       /* Rscan, max line size*/
#define REA_SIZE     10000      /*並び替え用*/
#define    CHR_SIZE  1024
#define C_SIZE       3          /*column size*/
#define D_SIZE       0.5        /*並び替え用閾値*/
#define    RR_SIZE   1000       /*max row size(最小二乗法に用いる点数)*/

double     datx[FRA_SIZE][LIN_SIZE];
double     daty[FRA_SIZE][LIN_SIZE];
double     datz[FRA_SIZE][LIN_SIZE];
double     dist;
int                fp[FRA_SIZE];
double     rx[REA_SIZE],ry[REA_SIZE],rz[REA_SIZE],rd[REA_SIZE];

main(argc, argv)
int argc;
char *argv[];
{
          FILE       *fpi, *fpo;
          char       ibuf[CHR_SIZE];
          int                i, j, k, n, m, o;
          int                di, rgb[3];
          int                l_size, lin, fra, lin_p, fra_p, r_size;
          int                ap, bp, cp, dp;
          double     a[RR_SIZE*3][C_SIZE], b[RR_SIZE*3];
          double     aa[C_SIZE][C_SIZE], bb[C_SIZE], x[C_SIZE];
          double     da, db, dis;
          double     sx, sy, sz;
          double     dx, dy, dz;
          double     dbox, xbox, ybox, zbox;
          double     min, f_size;
          double     xsum, ysum, zsum;
          double     xave, yave, zave;

          if(argc != 9){
            printf("--- Convert for Point to Surface Pro ---\n");
            printf("Usage:%s [fpi.txt] [lin_size] [sx] [sy] [sz] [f_size] [r_size] [fpo.txt]\n",argv[0]);
                exit(1);
          }

          if((fpi = fopen (argv[1],"rb")) == NULL){
                fprintf(stderr,"Can not open %s.\n",argv[1]);
                exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpi);

          if((fpo = fopen (argv[8],"w")) == NULL){
                fprintf(stderr,"Can not open %s.\n",argv[8]);
                exit(1);
          }
          fprintf(stderr, "Open status %2d \n",fpo);

          sscanf(argv[2], "%d", &l_size);
          sscanf(argv[3], "%lf", &sx);
          sscanf(argv[4], "%lf", &sy);
          sscanf(argv[5], "%lf", &sz);
          sscanf(argv[6], "%lf", &f_size);          /*閾値(面~点の距離)*/
```

```
	sscanf(argv[7], "%d", &r_size);			/*row size(最小二乗法に用いる点数)*/

	if(l_size>LIN_SIZE){
		fprintf(stderr, "too long size, LIN_SIZE.\n");
		exit(1);
	}
	if(r_size>RR_SIZE){
		fprintf(stderr, "too long size, RR_SIZE.\n");
		exit(1);
	}

	min = 1.0;
	n=0;	o=0;
	lin=0;	lin_p=0;
	fra=0;	fra_p=0;
	while (fgets(ibuf, CHR_SIZE, fpi) != NULL){
		if(n>0){
			sscanf(ibuf, "%lf %lf %lf %d %d %d %d", &datx[fra][lin], &daty[fra][lin],
&datz[fra][lin], &di, &rgb[0], &rgb[1], &rgb[2]);
			if(di>0){
				dx = sx-datx[fra][lin];
				dy = sy-daty[fra][lin];
				dz = sz-datz[fra][lin];
				dist = sqrt(dx*dx+dy*dy+dz*dz);
				if(dist <= D_SIZE){
					if(dist < min){
						min = dist;
						lin_p = lin;
						fra_p = fra;
					}
					rx[o] = datx[fra][lin];
					ry[o] = daty[fra][lin];
					rz[o] = datz[fra][lin];
					rd[o] = dist;
					o++;
					if(o>=REA_SIZE){
						fprintf(stderr,	"too	long	size,
REA_SIZE.\n");
						exit(1);
					}
				}
			}
			lin++;
			if(lin==l_size){
				lin=0;
				fra++;
				if(fra>=FRA_SIZE){
					fprintf(stderr, "too long size, FRA_SIZE.\n");
					exit(1);
				}
			}
		}
		n++;
	}
	fprintf(stderr, "line : %d\tframe : %d\n", l_size, fra);
	fprintf(stderr, "scaning count : %d\n", n-1);
	fprintf(stderr, "o : %d\n", o);
	if(o<r_size){
		fprintf(stderr, "too long size, D_SIZE\n");
		exit(1);
	}
	fprintf(stderr, "lin_p : %d\tfra_p : %d\tmin : %lf\n", lin_p, fra_p, min);

	/* rearrange */
	for(i=1; i<o; i++){
		for(j=i; j>0; j--){
			if(rd[j]<rd[j-1]){
				dbox = rd[j];
				rd[j] = rd[j-1];
				rd[j-1] = dbox;

				xbox = rx[j];
				rx[j] = rx[j-1];
```

```
                                        rx[j-1] = xbox;

                                        ybox = ry[j];
                                        ry[j] = ry[j-1];
                                        ry[j-1] = ybox;

                                        zbox = rz[j];
                                        rz[j] = rz[j-1];
                                        rz[j-1] = zbox;
                                }
                        }
                }
/*
                for(i=0; i<r_size; i++){
                        fprintf(stderr, "%lf\t%lf\t%lf\n", rx[i], ry[i], rz[i]);
                }
*/
                /*最小二乗法*/
                for(i=0; i<C_SIZE*r_size; i++){
                        a[i][0] = 0.0;          a[i][1] = 0.0;          a[i][2] = 0.0;
                        b[i] = 0.0;
                }
                m=0;
                for(i=0; i<r_size; i++){
                        a[m+0][0] = rx[i]*rx[i];
                        a[m+0][1] = rx[i]*ry[i];
                        a[m+0][2] = rx[i]*rz[i];
                        a[m+1][0] = rx[i]*ry[i];
                        a[m+1][1] = ry[i]*ry[i];
                        a[m+1][2] = ry[i]*rz[i];
                        a[m+2][0] = rx[i]*rz[i];
                        a[m+2][1] = ry[i]*rz[i];
                        a[m+2][2] = rz[i]*rz[i];
                        b[m+0] = rx[i];
                        b[m+1] = ry[i];
                        b[m+2] = rz[i];
                        m = m+C_SIZE;
                }

/*
                for(i=0; i<r_size*C_SIZE; i++){
                        fprintf(stderr, "%10.3lf\t%10.3lf\t%10.3lf\t|\t", a[i][0], a[i][1], a[i][2]);
                        fprintf(stderr, "%10.3lf\n", b[i]);
                }
*/
                /*正方行列に変換*/
                fprintf(stderr, "\na[%d][%d] * a[%d][%d] = a[%d][%d]\n", C_SIZE, r_size, r_size, C_SIZE, C_SIZE,
C_SIZE);
                for(k=0; k<C_SIZE; k++){
                        for(i=0; i<C_SIZE; i++){
                                aa[k][i] = 0.0;
                                for(j=0; j<r_size*C_SIZE; j++){
                                        aa[k][i] += a[j][k] * a[j][i];
                                }
                        }
                        bb[k] = 0.0;
                        x[k] = 0.0;
                        for(j=0; j<r_size*C_SIZE; j++){
                                bb[k] += a[j][k] * b[j];
                        }
                }
/*
                for(i=0; i<3; i++){
                        fprintf(stderr,  "%.3lf\t%.3lf\t%.3lf\t|\t%.3lf\n",  aa[i][0],  aa[i][1],  aa[i][2],
bb[i]);
                }
*/
                gauss(aa, x, bb, C_SIZE);

                for(i=0; i<3; i++){
                        fprintf(stderr, "x[%d] = %10.8lf\n", i, x[i]);
                }
```

```
	/* 同一面の判別(指定点から外側へ) */
	ap = 0;	bp = 0;
	cp = 0; dp = 0;
	xsum = 0.0;		ysum = 0.0;		zsum = 0.0;
	fprintf(fpo, "x\ty\tz\t%s\n", argv[8]);
	for(i=fra_p; i>=0; i--){
		fp[i] = 0;
		for(j=lin_p; j>=0; j--){
			if(datx[i][j]!=0.0 || daty[i][j]!=0.0 || datz[i][j]!=0.0){
				dis = 0.0;
				da = x[0]*datx[i][j]+x[1]*daty[i][j]+x[2]*datz[i][j]-1;
				db = sqrt(x[0]*x[0]+x[1]*x[1]+x[2]*x[2]);
				if(da<0){
					da = da*-1.0;
				}
				dis = da/db;
				if(dis<f_size){
				  xsum += datx[i][j];
				  ysum += daty[i][j];
				  zsum += datz[i][j];
				    fprintf(fpo,	"%lf\t%lf\t%lf\n",	datx[i][j],	daty[i][j],
datz[i][j]);
				  fp[i]++;
				  ap++;
				}
			}
		}
		if(fp[i]==0)
			break;
	}
	for(i=fra_p; i>=0; i--){
		fp[i] = 0;
		for(j=lin_p+1; j<l_size; j++){
			if(datx[i][j]!=0.0 || daty[i][j]!=0.0 || datz[i][j]!=0.0){
				dis = 0.0;
				da = x[0]*datx[i][j]+x[1]*daty[i][j]+x[2]*datz[i][j]-1;
				db = sqrt(x[0]*x[0]+x[1]*x[1]+x[2]*x[2]);
				if(da<0){
					da = da*-1.0;
				}
				dis = da/db;
				if(dis<f_size){
				  xsum += datx[i][j];
				  ysum += daty[i][j];
				  zsum += datz[i][j];
				    fprintf(fpo,	"%lf\t%lf\t%lf\n",	datx[i][j],	daty[i][j],
datz[i][j]);
				  fp[i]++;
				  bp++;
				}
			}
		}
		if(fp[i]==0)
			break;
	}
	for(i=fra_p+1; i<fra; i++){
		fp[i] = 0;
		for(j=lin_p; j>=0; j--){
			if(datx[i][j]!=0.0 || daty[i][j]!=0.0 || datz[i][j]!=0.0){
				dis = 0.0;
				da = x[0]*datx[i][j]+x[1]*daty[i][j]+x[2]*datz[i][j]-1;
				db = sqrt(x[0]*x[0]+x[1]*x[1]+x[2]*x[2]);
				if(da<0){
					da = da*-1.0;
				}
				dis = da/db;
				if(dis<f_size){
				  xsum += datx[i][j];
				  ysum += daty[i][j];
				  zsum += datz[i][j];
				    fprintf(fpo,	"%lf\t%lf\t%lf\n",	datx[i][j],	daty[i][j],
datz[i][j]);
				  fp[i]++;
```

```
                                        cp++;
                                }
                        }
                }
                if(fp[i]==0)
                        break;
        }
        for(i=fra_p+1; i<fra; i++){
                fp[i] = 0;
                for(j=lin_p+1; j<l_size; j++){
                        if(datx[i][j]!=0.0 || daty[i][j]!=0.0 || datz[i][j]!=0.0){
                                dis = 0.0;
                                da = x[0]*datx[i][j]+x[1]*daty[i][j]+x[2]*datz[i][j]-1;
                                db = sqrt(x[0]*x[0]+x[1]*x[1]+x[2]*x[2]);
                                if(da<0){
                                        da = da*-1.0;
                                }
                                dis = da/db;
                                if(dis<f_size){
                                  xsum += datx[i][j];
                                  ysum += daty[i][j];
                                  zsum += datz[i][j];
                                    fprintf(fpo,   "%lf\t%lf\t%lf\n",   datx[i][j],   daty[i][j],
datz[i][j]);
                                  fp[i]++;
                                  dp++;
                                }
                        }
                }
                if(fp[i]==0)
                        break;
        }
        fprintf(stderr,"\n");
        fprintf(stderr,"cp : %d | ap : %d\n", cp, ap);
        fprintf(stderr,"-----------------\n");
        fprintf(stderr,"dp : %d | bp : %d\n", dp, bp);

        xave = xsum/(ap+bp+cp+dp);
        yave = ysum/(ap+bp+cp+dp);
        zave = zsum/(ap+bp+cp+dp);
        //fprintf(fpo, "%lf\t%lf\t%lf\n", xave, yave, zave);

        fclose(fpi);
        fclose(fpo);
        exit(0);
}

gauss(a, x, b, n)
double  a[C_SIZE][C_SIZE];
double  x[C_SIZE];
double  b[C_SIZE];
int             n;
{
        int             i, j, k;
        double  p, g, s;

        for(k=0; k<n-1; k++){
                p = a[k][k];
                for(j=k+1; j<n; j++){
                        a[k][j] /= p;
                }
                b[k] /= p;
                for(i=k+1; i<n; i++){
                        g = a[i][k];
                        for(j=k+1; j<n; j++){
                                a[i][j] -= g * a[k][j];
                        }
                        b[i] -= g * b[k];
                }
        }
        x[n-1] = b[n-1] / a[n-1][n-1];

        for(k=n-2; k>=0; k--){
```

```
                    s = b[k];
                    for(j=k+1; j<n; j++){
                            s -= a[k][j] * x[j];
                    }
                    x[k] = s;
            }
}
```